Helsinki School of Economics

Information and Service Management

RISK MANAGEMENT IN SOFTWARE DEVELOPMENT PROJECTS

MASTER THESIS

Author: Alina Horodniceanu

Supervisor: Prof. Virpi Kristiina Tuunainen

Approved by the head of the dept of Management
24.11.2004 and awarded the grade excellent,
80 points.
Virpi Tuunainen and Petri Hallikainen

Abstract

In the strong contest of the high technology industry, the capability to develop fast new products is today a necessity of thriving or even survival for organizations. Technology is advancing all the time with an accelerating pace, enabling more sophisticated solutions and products, whereas customers are demanding them accordingly.

Managing of software projects is especially challenging, as software is not a physically measurable element. Successful project management requires a large variety of skills, and there are plenty of process areas that must also be involved. Over the past years, literature and practice recorded significant contributions to risk management field, the latter being considered an important fraction in software development.

This thesis emphasizes the recent idea according to which risk management has been identified as one of principal drivers of software development project success. The area of risk management was closely correlated to the project management field applied in software development projects. The thesis describes the essential areas of risk management, both in general and especially from software projects' point of view. The intention of this study is to present and analyse the best practices applied for risk management in software projects. We have chosen to explore Nokia Networks' software development processes with the help of a case study.

Based in principal on qualitative research methods and action research, but also reflection of the author, the thesis presents some possible ways to improve software project management by means of risk management practices.

Keywords: risk, risk management, project, project management, software development project, project management.

ACKNOWLEDGEMENTS

This thesis was written during year 2004 in Nokia Networks, Helsinki site. I want to thank all the people who contributed to the completion of this study, especially my supervisor from Helsinki School of Economics, Prof. Virpi Kristiina Tuunainen and Assistant Professor Petri Hallikainen, who gave me valuable feedback and all my colleagues from Nokia Networks who provided me the help and resources to finish this work. I am mostly grateful to my husband and my family who supported me throughout the hard hours of working and studying.

I wish to dedicate this work to the memory of my father, Eng. Constantin Cornel Horodniceanu, without whose words of advice I would have never achieved any of the things I am now proud of.

Helsinki, October 2004

Alina Horodniceanu

Table of Contents

1. INTRODUCTION

1.1 Background and Motivation

The interest aroused in the area of software development projects leaded me to the attractive field of risk management. My opinion is that there are obvious correlations between applying in practice the principles of risk management and the success of a software project. We believe that many software companies are at the beginning of establishing a thorough risk management strategy, especially as software is playing an ever-increasing role in today's society and industry.

Analyzing retrospectively success or failure of software project, one also sees the importance of skilled project management, particularly because modern software organizations operate in a highly dynamic market, under tight time and cost constraints. As an answer to these business and market needs, organizations have started to invest in developing software processes aiming to increase the efficiency of software development process, the software maturity and software product quality. By gaining experience in process development improvements, it has been shown that benefits and success have been obtained.

More and more companies and software projects today are relying on virtual teams to get things done. Through virtual team, it is addressed the problem of managing large software projects to improve the economics of their planning and execution. This is a critical problem that can radically benefit from the rapid emergence of the Internet and other global networks and their role as a global dis-intermediated service market place. In this work, we will only refer to the virtual teams located at different sites from the same organization. Virtual projects or sub-projects may provide a number of benefits and in the same time, may raise a number of problems and risks.

We believe that through a good management and coordination of all the areas presented above, software products could be brought to market faster, with high quality, fair price and in a controlled way.

1.2 Risk and Risk Management Concepts

Increasing competition, more demanding customers, the increasing pace of technological development and other changes, increasing complexity and novelty of business opportunities, are all demanding the need for more structured, systematic and effective approach to managing uncertainty, project and business risks.

According to Software Engineering Institute (SEI), risk is the possibility of suffering loss and in a development project; the loss describes the impact to the project which could be in the form of diminished quality of the end product, increased costs, delayed completion, or failure. Van Scoy argues that "Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity"(Van Scoy, 1992).

In Software engineering, there have been identified two categories of risk management: continuous risk management and non-continuous risk management.

Continuous Risk Management is a software engineering practice with processes, methods, and tools for managing risks in a project. It provides a disciplined environment for proactive decision-making to:

- Assess continuously what can go wrong (risks).
- Determine what risks are important to deal with.
- Implement strategies to deal with those risks.

In Continuous Risk Management, risks are assessed continuously and used for decision-making in all phases of a project. Risks are carried forward and dealt with until they are resolved or they turn into problems and are handled as such. Non-continuous Risk Management implies that risks are assessed only once during initial project planning. Major risks are identified and mitigated, but risks are never explicitly looked at again.

There are seven risk management principles, which provide a framework to accomplish effective risk management.

Global perspective	• Viewing software development within the context of the larger systems-level definition, design, and development. • Recognizing both the potential value of opportunity and the potential impact of adverse effects.
Forward-looking view	• Thinking toward tomorrow, identifying uncertainties, anticipating potential outcomes. • Managing project resources and activities while anticipating uncertainties.
Open communication	• Encouraging free-flowing information at and between all project levels. • Enabling formal, informal, and impromptu communication. • Using processes that value the individual voice (bringing unique knowledge and insight to identifying and managing risk).
Integrated management	• Making risk management an integral and vital part of project management. • Adapting risk management methods and tools to a project's infrastructure and culture.
Continuous process	• Sustaining constant vigilance. • Identifying and managing risks routinely through all phases of the project's life cycle.
Shared product vision	• Mutual product vision based on common purpose, shared ownership, and collective communication.

	• Focusing on results.
Teamwork	• Working cooperatively to achieve common goal. • Pooling talents, skills, and knowledge.

Table 1. Seven principles of risk management

Overall, we may say that Risk Management is a discipline that enables people and organizations to cope with uncertainty by taking steps to protect its vital assets and resources. It is not just about identifying risks; it is about learning to weigh various risks and making decisions about which risks deserve immediate attention.

1.3 Objectives of This Thesis

The primary objective of this study is to analyze how a comprehensive risk management strategy can influence the success of software development projects. Therefore, at general level, we want to study the risk management process and practices as constitutive part of project management of software projects.

This objective can be divided in several sub-objectives and we can divide the sub-objectives in theoretical and empirical. The theoretical sub-objectives we have followed are:

- Study and correlate the theories related to software projects and risk management;
- Find the place of risk management in project management processes;
- Based on the models already existing, create a model/framework containing the key parts of risk management

The empirical sub-objectives we have tried to achieve in the following chapters are:

- Analyse and test the theoretical framework proposed with the help of a case project; represent the critical areas of software project management, based on everything described above, and give recommendations for what should be done next.

- Improve the theoretical framework with the conclusions drawn after the case study and create a proposal for a model.

The analysis of risk management through Utility Theory is out of the scope of this work. We recognized the importance of this theory and its practicability and consequently maybe can be further investigated in future studies.

1.4 Research Methods

The first and probably the most comprehensive part of the thesis is built up from literary research. I have studied a number of books, articles and other publications about risk management, software development and project management and comprised the essential subtext of them into this thesis.

The research method is based on qualitative research and particularly on action research. We relied on qualitative research approach because of several main reasons. First, we found it difficult to evaluate the model proposed in a quantitative experiment. Second, we have only used one case study and the quantitative methods could not apply. Third, this method was chosen because I considered this work meant to develop the quality of work inside the organization and its performance, being directly concerned to improve my own and my colleagues practice. Interviews with the people involved were also held on informal basis.

The results of this work are intended to be applicable to any kind of software development, as the application domain covered by the empirical study suggests. However, we believe that medium to large organizations and fairly complex projects are more likely to benefit from the results of this research.

As mentioned above, the first phase of this thesis is theoretical. This part was divided in two main sub-parts:

1. Software development projects and project management overview: models, processes, principal phases and participants. One chapter is dedicated to multi-distributed projects. They play nowadays an important role in software development and we believe that in large organizations very few software programs do not contain virtual teams. Our study case is focused on a software program, which includes a distributed project.

2. Risk Management overview: the literature in risk management is very large. We have tried to extract some of the concepts, models and framework that we found valuable for our study in software development projects.

The second phase of our research consists of creation of the basis on which we will conduct the empirical part. We have tried to assemble the key ideas of the theoretical part and create a starting point/model proposed for the case study. This is materialized in the integration of risk activities in software projects and identification the key risk activities identified for software projects.

The empirical part is a mean to analyze, evaluate and finally create an improved model for managing risks in software development projects. The action research methods were therefore used for this purpose.

1.5 Structure of the Thesis

The present thesis is structured in 7 chapters, the last one containing the conclusions we can draw from the findings of the work.

The first chapter presents briefly the intentions of the study, introducing us to the basic concepts of risk management and research methods. The literature review and research are spread throughout the next 2 chapters. Chapter 2 is an overview on the basic principles of software development projects, including references to development projects in general, then development process and models, being closed with a short presentation of projects in distributed environment. Chapter 3 presents the literature review related to Risk Management concept and the well-known frameworks in the field.

Chapter 4 explains the place that risk management takes in software projects and the framework we propose for study. This contains the most important activities that we believe should be included in any risk activities within projects.

Chapter 5 is the presentation of the case study. To be able to enter in the practical work inside one company, we have started by presenting shortly the processes followed inside Nokia Networks and then the project we evaluate.

Chapter 6 contains the findings of this thesis, with proposed diagram for risk management workflow.

2. SOFTWARE DEVELOPMENT PROJECTS – LITERATURE OVERVIEW

The following chapter deals with the basic concepts of software development processes and project management. A short opening of the role played by consistent risk management procedures in the success of a software development project is presented as an introduction to the following chapters, where we develop present the literature review on risk management and our risk management framework.

The development of software projects follows creation and development processes, well defined and established in software companies. The software products creation process provides a common and unified way of working through the company or business unit.

Understanding the importance of software processes is critically important for producing high-quality software within on time and budget. The software development process comprises software engineering activities, including technical and managerial ones that are carried out in the production of software. The benefits of these life cycle models may result in improved product quality, increased effectiveness of methods and tools, reduced software development and maintenance costs, and increased customer and developer satisfaction. (Madhavji 1997)

Different processes decompose activities in different ways. The timing of the activities varies, as do the results of each activity. Different types of product may be produced by an organization using different processes. However, some processes are more suitable than others for some types of application. If the wrong process is used, this will probably reduce the quality or the usefulness of the software product to be developed. (Sommerville 1996)

Zahran (1997) defines the need of process management in his article: Without defined common processes that project members follow to perform their tasks, management will not be able to properly measure the progress of the project. There are a number of key processes, which are the most critical for managing a software project. These processes, if documented, agreed, and followed, will lead to the success of a SW project. Zahran (1997) has divided the processes into two

groups. The first group includes the process, which focuses on the overall management control, while the second group includes the processes necessary to ensure co-ordination between the development activities.

Management and Control Processes (Zahran 1998):

• Software Requirement Management;

This process aims to ensure that the requirements are controlled and that they form a baseline for the project management plans and software development activities.

• Software Project Planning;

This process aims to ensure that software estimates and uncertainties are documented for use in planning and tracking the software project.

• Software Risk Management;

This process aims to ensure that risks are identified, qualified, and managed continuously throughout the development phases of the project.

• Software Project Tracking;

This process aims to ensure that realistic goals are set up for the project's cost, time and technical features.

• Software Subcontract Management;

This process aims to ensure the selection of a qualified software supplier, agreeing with the supplier the mutual commitments, tracking and reviewing the subcontractor's performance and results, maintaining communications with the supplier and tackling the issues as they arise.

Co-ordination Processes (Zahran 1998):

• Software Quality Assurance;

This process aims to ensure that software quality assurance activities are planned and executed in order to verify the conformance of software products to

applicable standards, and elevate non-compliance issues to senior management when necessary.

• Software Configuration Management;

This process aims to ensure that software configuration and release management activities are performed, ensures that selected software products are identified, that changes to the identified items are controlled, and that changes to the baseline items are disseminated to affected groups and individuals.

Having assessed these processes, a software project has its foundation ready. The choice for one particular type of software development process depends on the type of application to be developed and the amount of risks carried by introducing it to the market.

There are three software processes models currently used in practice: Waterfall model, Iterative and Spiral Model. For the sake of our dissertation, we will look closely in the next chapter at the Iterative Software Development Process, as the one to be investigated in our study case.

2.1 Incremental Development Model

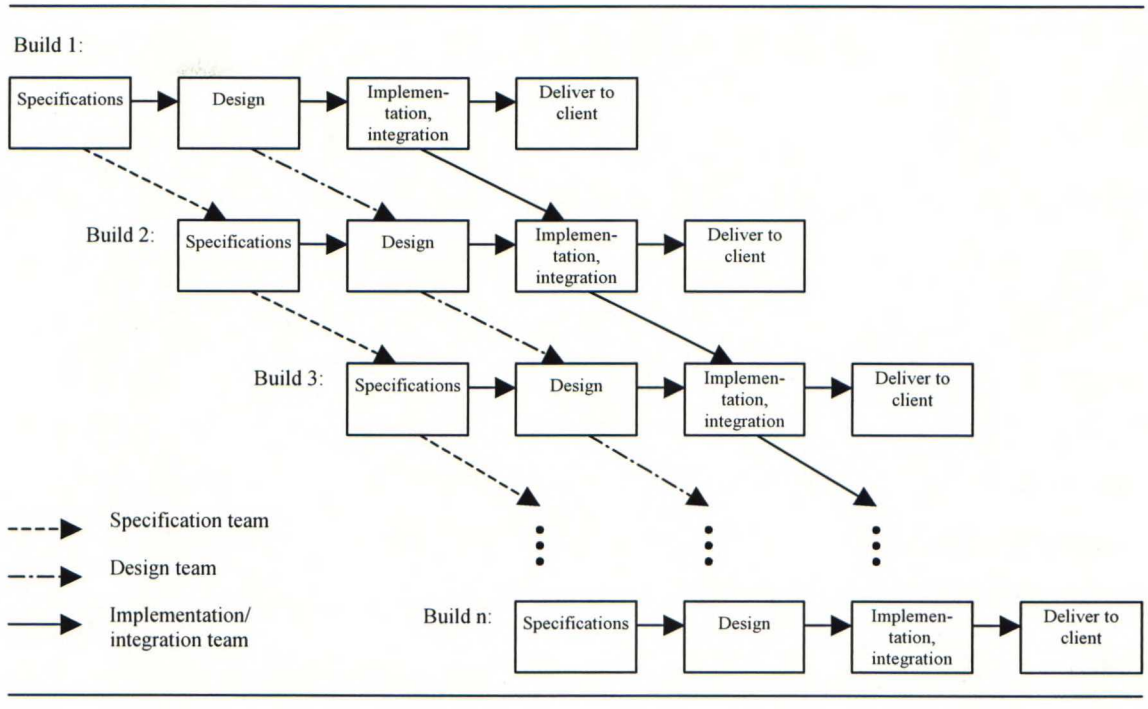

Figure 1. Incremental model. (Schach 1999)

Incremental process model is one of the evolutionary process models and it has an iterative philosophy, which means that it is not necessary to fully understand all customer requirements before the development of the product is started. At the beginning only the most important core features of the product must be defined but the less important feature definitions are made more accurate along the way.

The whole software product is divided into *increments*, which could be considered as subprojects. Every increment contains a small part of product's functionality. The incremental process consists of small waterfalls (increments) with the idea that each increment produces a part of the software. Incremental model delivers an operational product with every increment. In each increment some functionality is added to the software until after the last increment the whole system is ready. The output of the first increment is often the core product, which includes only the basic requirements. The customer can try it out, evaluate it and suggest changes and improvements to the software. On the basis of customer evaluation a plan is

developed for the next increment. The plan might contain changes to the core product or some additional features. (Pressman 2000)

Basic strategy for incremental development is to divide the product into small manageable steps: 1) Plan a little 2) Specify, design and implement a little 3) Integrate, test and run little. Each increment is kind of a small project with internal structure of the waterfall model. In each increment requirements for that increment are gathered and analyzed, part of the product is designed, implemented and tested. Incremental model accommodates changing customer requirements by creating the product little by little; changes in customer requirements (for example additional functionality) can be implemented in later increments. (Jacobson 1998) Incremental model is basically a development of waterfall model with a purpose to allow specification changes during development and to provide some needed stability to the development process. (Sommerville 1996a)

The lifecycle of the project is made of increments. The *early increments* help to understand the risks, establish feasibility, build the initial core of the software (internal preliminary releases) and make the business case. *Later increments* then add functionality to the product until external release is ready. Although an increment is a "miniproject" it is not totally independent but must follow the guidelines and schedules set by the project. (Jacobson 1998)

Incremental development avoids the problems associated with constant change. At the beginning of the process overall system architecture is established and increments are then developed according to this architecture description. As the products of an increment are developed, tested and approved, they are not changed unless errors are detected or customer asks for changes. The interfaces of a delivered increment are frozen and later increments must adapt to those. Documentation and plans must be done for each increment. These facts make incremental development more manageable than other evolutionary approaches. (Sommerville 1996)

How to divide the whole product into small increments? First, a high-level analysis must be done to divide the product into "slices". Each slice should be executable, so it can be tested against the requirements. A slice should cut across as much of the functionality of the system as possible. This means that a slice should be

more like a use case or a part of a use case rather than a single subsystem. It should be functioning entity, executable, something that can be shown to a user. It should represent a product feature. If parts of the project must be eliminated (some features removed) in some phase (e.g. to meet the schedule), it is easier to remove whole slices rather than parts of multiple slices. It is the task of project managers to choose which slices are implemented first. To gain benefit from incremental model the most risky slices, which are most prone to errors, should be implemented first. This way the major problems, which can cause failure of the whole project, are found as soon as possible. (Martin 1999b)

2.1.1 Strengths of the Incremental Development Model

Early customer feedback

Critical functionality is available early. Customers waiting for most important functionalities don't have to wait until the whole product is ready. (McConnell 1998)

It is easier for customers to understand a system that actually works than just a pile of documentation. Customers can operate the preliminary releases of the system and provide project team with additional or changed requirements and improvement suggestions. Changes can easily be implemented in next increment. (Jacobson 1998)

Incremental development enables acquiring customer feedback in early phases of the process. It also limits system errors as the development team is concerned with only one part of the software system at a time. (Sommerville 1996)

Using series of increments offers a way to add user experience to development of the final product. This represents iterative enhancement of the final product. (Sodhi 1991)

Flexibility

Incremental development model allows starting the project even if only few resources are available. It is possible to implement the first increments with fewer people and add more people to the project later. (Pressman 2000)

Training new people is easier because they can be trained on the work. Special trainings to help people just to understand what the process is are unnecessary. When working with someone who has done it before they soon learn the working practices. After a couple of increments everyone understands the process and workflows. As incremental project proceeds, initial small team gets familiar with new technologies, tools and processes. When project team grows step-by-step, the core team can teach new members. Core team can fine-tune the process and tools before most people enter the project. (Jacobson 1998)

Incremental process structure requires the product architecture to be open for modifications. This openness can be a burden in development phase while development is somewhat dependent on clear specifications and architecture, but it is a great advantage in later phases of the product lifecycle. Products developed with waterfall process's coherent and cohesive design will probably work well as long as no significant modifications are done to the product. In maintenance phase, however, modifications must be made every now and then and if product's design does not accommodate enhancements it is possible that a large proportion of the product must be rebuilt in order to hold the product together. In incremental model the openness of the design adapts changes easily. Modifications made in maintenance phase are no different to modifications made in development phase when adding new functionality to existing product. In other words, any modification needed in maintenance phase can be considered as a new increment. "If a design is flexible enough to support the incremental model, then it will certainly allow virtually any kind of maintenance without falling apart." (Schach 1999)

Communication between phases

The product is developed in a series of increments. This means that for example specification phase does not need to be completed in one go: For an

increment a part of the system is specified. When this part enters implementation phase, some deficiencies might be detected in the specifications. Feedback about these deficiencies is given to the specification responsible. When specifications of next increment are made, they can be made better according to the feedback received. The same idea works with all phases.

Reduced risks

Integration is done in each increment, which reduces technical risks, such as incompatible software and hardware. Requirements risk is reduced because customer gets usable software as soon as possible and can test it. Therefore misconceptions in requirement analysis phase become evident much sooner than in the traditional waterfall model. Making long-perspective plans is inaccurate and often impossible. In incremental development model plans can be revised separately for each increment so there is no need to make detailed plans for the whole project at the beginning of the project. This means that planning risk is also reduced. (McConnell 1998)

Iterative development reduces serious risks in early increments. In waterfall model serious risks are not addressed until the integration and testing phases where problems start to explode all at once. (Jacobson 1998, p.90)

More accurate estimations and learning from mistakes

Making estimations is easier. Instead of having to estimate the whole project at once, smaller estimates can be made separately for each release. It is also possible to learn from bad estimates during the project: one can continuously enhance the accuracy of estimates by observing the success of estimates in previous increments and adjusting estimation methods accordingly for following increments. (McConnell 1998)

Continuous and early integration

In incremental development model problems and faults are uncovered in a steady flow because integration and testing is done in every increment. It is easier

to handle this kind of steady flow of faults than the sudden discovery of large amount of faults, which happens in the integration phase of waterfall model. (Jacobson 1998) Large amount of faults is found at once in a waterfall project and simultaneous schedule pressure leads to quick fixes of poor quality. Schedule of waterfall model project is often delayed when this happens. Integration testing can start early in incremental model because product of each increment is a working whole.

Short time to market

Incremental approach is probably best in situations where schedule is so tight that full functionality cannot be delivered reasonably in time. Incremental development allows the customers to get their hands on the core product early. (Pressman 2000)

Product can be introduced gradually to customer organization. After each increment customer receives a functioning product and is able to do some useful work with it. It is not necessary to wait until the whole product is completed. Customer has more time to get familiar with the product and train personnel to use new system. (Schach 1999)

Visible progress

Project team delivers a working product with gradually increasing functionality at the end of each increment. It is easy for customers and other stakeholders to see that project is progressing when compared to waterfall model, in which all that exists in early phases of the project is documents. (Jacobson 1998)

Problems with the project progress become evident early because it is easy see whether an increment release is delivered in time or not. Project progress reporting is less time-consuming because "the working software is a more accurate status report than any paper report could ever be". (McConnell 1998)

2.1.2 Weaknesses of the Incremental Development Model

Increasing project control costs

Incremental development increases project's overhead costs. More time is spent on retesting in each increment, version control tasks, increment planning and supporting numerous versions of software releases. (McConnell 1998)

Increased number of releases

Incremental development has some drawbacks. When customer receives a new version of the product, the time of the project team goes to solving and correcting errors with the customer and the development of the next increment is stopped. This is a challenge for project management. It must be kept in mind that the use of too small increments can lead to corruption of the software architecture. (Haikala 2000)

Appropriate documentation of every version of the product is not cost-effective if the number of releases is high. This might hinder management from getting needed deliverables for project progress measurement. (Sommerville 1996) Increasing number of document versions means that more time is needed for approving documents. This may cause slipping from formal approval procedures in order to save time. (Sommerville 1996) Each increment must be integrated into the existing product structure without destroying anything that has already been built. (Schach 1999)

Difficulties in keeping it all together

Incremental development project requires careful planning at the beginning of the project. Contents of the project must be divided into increments with care and the order of the increments must also be reasonably planned. The key criterion for a good plan is that after any increment the system, although possibly incomplete, is always a working wholeness. (Jakobsen 1998)

The use of incremental model requires thinking the product as a whole right from the beginning of the project in order to create a design that supports the whole product. In the same time the product has to be considered as a series of increments, each having independent contents and requirements. This requires a skilled developer. If these contradictory views on the product cannot be appropriately handled, the use of incremental process may lead to unsatisfactory results; if control of the process is lost, the project can end up with a product, which is far from open-ended by terms of design and architecture and thus hard to maintain. (Schach 1999)

Incomplete requirements at the beginning of the project

At the beginning of an incremental project general level system architecture must be established in order to plan the contents of the increments. However, at the beginning of the project the requirements are incomplete. Decisions have to be made on the basis of the known requirements. This can cause the requirements to be constrained by the architecture that is established. (Sommerville 1996)

More complex interdependencies

In incremental model the dependencies between process activities and work-products are more complex than in waterfall model. The amount of resources is finite and various activities are competing for those resources. If work demands are inconsistent, it can lead to peaks and troughs in resource usage. Another consequence of incremental model is changing work-products. Any change made to the work-products in one phase can cause re-work in later phases. A schedule delay or a change in the quantity of work in any increment has potential to cause dramatic over-allocations of team resource which, in turn, can delay or impact on the quality or performance of later phases. This kind of dynamic behavior of incremental development model sets new challenges on planning, control, and improvement activities of the project. (Powell 1998)

2.2 Software Project Management

Choosing the software development process is perhaps one of the most important decisions to be made by the project management team. The reason behind this affirmation comes from the nature of the software product, which is troublesome when considering it from the project management's point of view. When compared to other, more "traditional" type of engineering projects, for example bridge building, software project management is more difficult because of a number of reasons (Sommerville 1996):

- The completeness of the product is hard to be assessed and effects of schedule delays are not visible. The only way software manager can control progress is by relying on the documentation produced by other individuals.

- Software development is quite new business. There is no standard process; the software development process is not well understood, tried and tested. The relation between software process and product type is still unclear. It is hard to know what kind of process will work well with a given product.

We may consider therefore that the software projects are characterized by uniqueness. New systems often differ considerably from the ones developed in past projects. Technology advances rapidly and new kind of skills must be adapted accordingly. The lack of previous experience makes it very hard to evaluate uncertainties related to development projects.

The software development cycle includes many steps. Some of the steps have to be repeated until the system is complete and the customer and users are satisfied (Pfleeger 1998). However, ongoing processes need support from resources, which are installed for reaching a specific objective within certain specifications. The same process may be repeated with quite similar steps, but requirements and the goal that needs to be obtained may vary.

Projects need processes to manage day-to-day activities and to achieve the desired outcome and performance level.

2.3 Processes and Software Projects

This chapter considers basic software concepts and it gives a more detailed explanation of overall project management processes. At first, the key components that lead to effective software project management are presented and distinct characteristics between processes and projects are pointed out. Because of the purpose of this study, only a little attention is given for the area of cost management and leadership skills as well as resource management, even if they are important issues of project management.

Processes and projects differ mainly because processes are ongoing and repetitive by their nature while projects are brief and unique.

Project Management Institute (www.pmi.org), a leading professional association in project management (PMI) (2000, p. 4) has identified distinctive characteristics of projects: "A project is a temporary endeavor undertaken to create a unique product or service. Every project has a definite beginning and a definite end. A project produces a product or service that is different in some distinguishing way from all other products or services." Projects are undertaken at all levels of the organization. They may involve a single person or many thousands of people. Duration or projects ranges from a few weeks to more than five years. Projects may involve a single unit of one organization or they may cross-organizational boundaries, as joint ventures and partnering. (PMI 2000, p. 4) PMI (2000, p. 10) brings up also certain types of endeavors, which are closely related to projects.

For example, *a program* is a group of projects managed in a coordinated way to obtain benefits not available from managing them individually. Many programs also include elements of ongoing operations. In some application areas, program management and project management are treated as synonyms; in others, project management is a subset of program management. Because of this classification, there is still one important project management term, which needs a clear definition. Projects are frequently divided into more manageable components or *sub-projects*. Sub-projects are often contracted to an external enterprise or to another functional unit in the performing organization. (PMI 2000, p. 4)

Managing a project is not an easy process. The management process involves handling and facilitating complex interactions between and within various groups of people who are directly or indirectly involved in the project and are interested in its successful conclusion. The skills needed for accomplishing this task vary as the needs and the activities of the project evolve and change during its lifetime. (Berkeley et al. 1990)

Kerzner (2003) gives an overview definition of project management: "Project management is the planning, organizing, directing, and controlling of company resources for a relatively short-term objective that has been established to complete specific goals and objectives".

Furthermore, project management is designed to manage or control company resources on a given activity, within time, within cost, and within performance. Also good customer relations can be considered as a fourth constraint of project management.

Quite many authors like Phillips (2000), Pressman (2001) and McConnell (1996) have divided software project management into three or four dimensions. They all have stated that balancing between 4P's - people, process, product and project (or technology) (McConnell 1996) - software can be developed and maintained successfully. It can be noticed that also other software development books are written based on these four dimensions, but more indirectly.

The *product* is the project's final outcome and products include software, documentation, and training and maintenance services (Phillips 2000). Before the project can be planned, the project team or representative and customer must meet to define product objectives and scope should be established, alternative solutions should be considered, and technical and management constraints should be identified (Pressman 2001).

WITHIN GOOD CUSTOMER RELATIONS

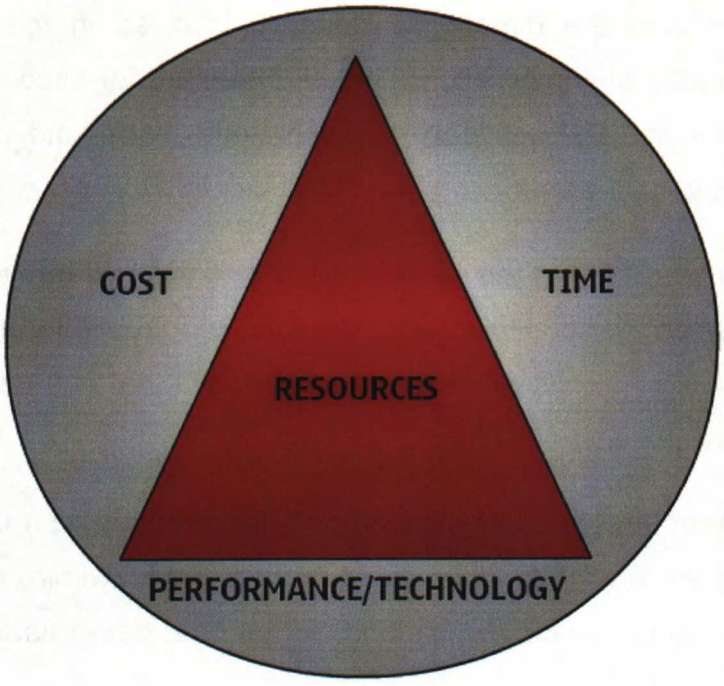

Figure 2. Project management overview (Kerzner 2003).

2.3.1 Overall Project Management Process

Project management processes describe, organize, and complete the work of the project. In addition, they are applicable to most projects and most of the time. Organization performing projects will usually divide each project into several project phases to improve management control and provide for links to the ongoing operations of the performing organization. Together, as a whole, project phases create the project life cycle (PMI 2001, p. 11, 30).

This chapter provides an introduction to the concept of project management as a number of interlinked processes and their interactions.

The project life cycle serves to define the beginning and the end of a project. It generally describes what technical work should be done in each phase (e.g. is the work of the architect part of the definition phase or part of the execution phase?) and who should be responsible of each phase (e.g. who needs to be

involved with requirements and design?). (PMI 2001, p. 12) The project life cycle is composed of project management processes. PMI (2001, p. 30) has classified processes into five groups.

The process groups are *initiating, planning, executing, controlling* and *closing* processes. The groups are linked by the results they produce; the result or outcome of one often becomes an input to another. The iterative links between the two groups in the middle of processes are fulfilled because planning provides both; executing with a documented project plan and then documented updates to the plan as the project progresses. Here, controlling processes include also maintenance, one very important task in project management. The groups are not discrete, one-time events; they are overlapping activities that occur at varying levels of intensity throughout each phase of the project. (PMI 2001, p. 30) But what should be done in each phase in a general level?

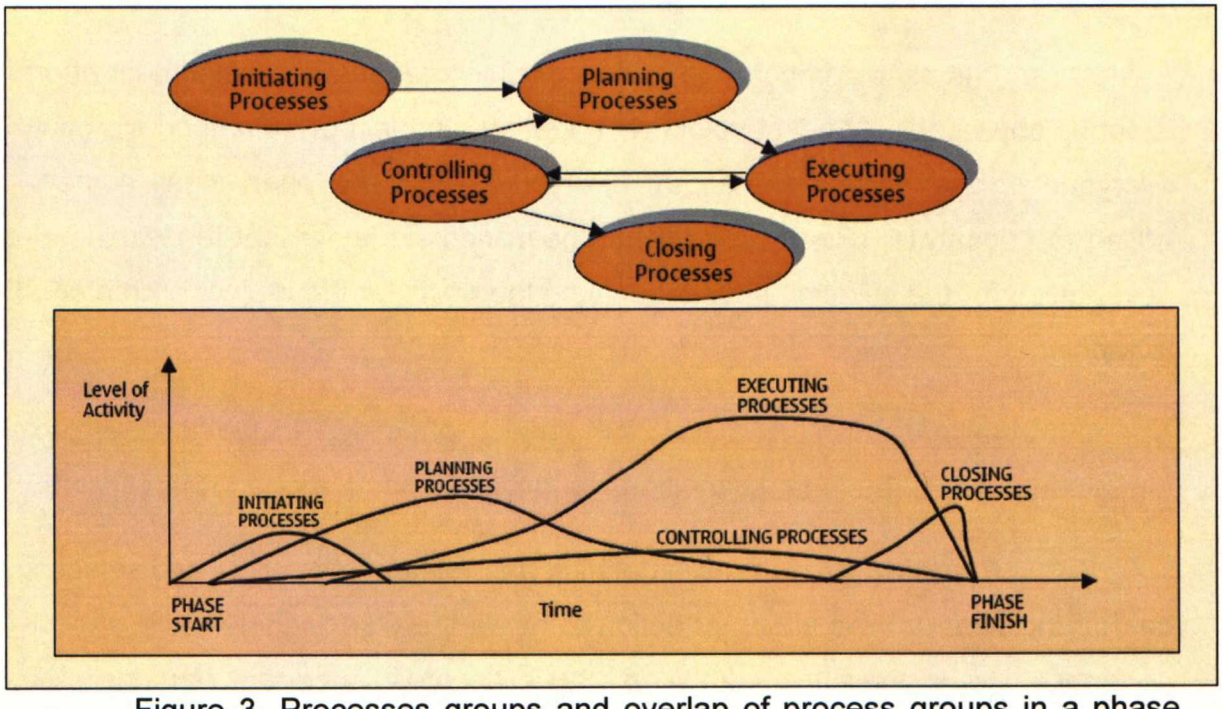

Figure 3. Processes groups and overlap of process groups in a phase (adapted from PMI 2001, p. 28-29).

Initiating processes – authorizing the project or phase (PMI 2001, p. 30). Each project begins with an idea, vision, or business opportunity, a starting point that must be tied to the organization's business objectives. Before any project could start, before it could move for planning phase, it needs to define what needs to be

accomplished and decide how the project is going to achieve those objectives. (Kerzner 2000) Projects are usually authorized as a result of one or more of the following: a market demand, a business need, a customer request, a technological advance or a legal requirement.

Different needs will cause problems, opportunities, or business requirements but the central theme for all these terms is that management generally must make a decision about how to respond. (PMI 2001, p. 48)

The project charter is the foundation of the project and it is a key output from initiation. It includes a document of a business need, an agreement on what the project is committed to deliver, an identification of project dependencies, the roles and responsibilities, and the standards for how project budget and project management should be approached. The project charter defines the boundaries of the project. Once the project boundaries are defined the planning phase could begin. (Kerzner 2000)

It is almost impossible to successfully manage a development effort that is set up improperly. The first objective in getting a project off to a good start is to get everyone on the same wavelength. Everyone has to be open to all comers and willing to cooperate. (Reel 1999). It has been noticed lately that too much value is never given for the initiation phase, because the basis for the project is created at the kick-point.

Initiation is the basis for project planning, including the project's fundamental goals, boundary conditions, and limitations. (Haikala & Märijärvi 1998)

Planning processes – defining and refining objectives and selecting the best of the alternative courses of action to attain the objectives that the project was undertaken to address (PMI 2001, p. 30). The most important responsibilities of project work are planning, integrating, and executing the plans. Because of relatively short duration of projects and prioritized control of resources, almost all projects require formal and detailed planning. The integration of planning should be done carefully because different functional units may develop their own planning documentation but anyhow, every plan should work well together. (Kerzner 2003) The project plan is the game plan for the project. Planning should be performed

before work begins on project activities. It should include both general project planning and more detailed planning for activities in the near future. (Miller 1997) As Thayer et al. (1997) have noted: "Planning is deciding in advance what to do, how to do it, and who is to do it".

The question is what are the elements of a good plan, what are the most essential areas to focus on. Planning is concerned with business goals. Solutions must serve the business in a timely and economic manner. They must accurately represent what the software people can do for the business. (Phillips 2000) Before any planning work can be started, the requirements should be captured and analyzed. Deciding precisely what to build and documenting the results is the goal of the requirements phase of SW development.

The activity after the SW requirements is the determination of SW scope. Software scope describes the data and control to be processed, function, performance, constraints, interfaces, and reliability. Things are always somewhat hazy at the beginning of a SW project. A need has been defined and basic goals and objectives have been enunciated but there is still a large degree of uncertainty inherent in planning. Estimation of resources, cost, and schedule for SW engineering requires experience, access to good historical information, and the courage to commit to quantitative predictions when qualitative information is all that exists. (Pressman 2000)

Project planning includes also other important management areas, which serve as support functions for the project planning. For example, risk management planning, communication planning, tools and methods to be used and quality assurance are the key facilitating areas during project planning and should be observed regularly. (Haikala & Märijärvi 1998)

Executing processes – Coordinating people and resources to carry out the plan (PMI 2001, p. 30). Execution of the plan, of course, is what a project is all about? When the agreement is signed and the project becomes a reality, it is time to move ahead, but what are the next steps?

During the project executing phase the primary process for carrying out the project plan is performed. The main part of the project budget will be expanded in

performing this process. In this phase, the project manager and the project management team must coordinate and direct the various technical and organizational interfaces that exist in the project. Whereas project members are using their skills and knowledge for carrying out the project plan by performing included activities and enhancing project performance. (PMI 2001, p. 34, 44)

Controlling processes – Ensuring that project objectives are met by monitoring and measuring progress regularly to identify variances from plan so that corrective action can be taken when necessary (PMI 2001, p. 30). Whitten (1995) has claimed that a software project is like a living organism and in order to survive, all its vital parts must function in harmony. If one of the parts fails to perform its mission, dependent parts will also begin to fail soon. Control of execution makes a project successful.

Identifying and correcting things that are not proceeding according to the plan achieve control. (Miller 1997)

Usually, the main problems are emerging in the controlling phase. The best action against problems is trying to take preventive actions and putting recovery plans in place before unrecoverable harm occurs. Easier said than done, but controlling the project is all about staying alert and making decisions in a reactive, dynamic environment. (Whitten 1995)

"It is fair to say that system's life does not end with delivery", claims Pfleeger (1998) in his book. The final system is usually subject to continuous change, even after it is built. System development is complete when the system is operational, that is, when users in an actual production environment are using the system. Any work done to change the system after it is in operation is considered to be maintenance. (Pfleeger 1998)

Today maintenance is thought of as continued development. SW maintenance is quite expensive because it requires that people work for years, it relies on people even more than development. Still, SW developers can help themselves by creating easier to maintain SW. (Phillips 2000)

Closing processes – Formalizing acceptance of the project or phase and bringing it to an orderly end (PMI 2001, p. 30). A project is a solitary series of activities, with limited duration, resources, and specified objectives; it must end at some point of time. To be able to close the project is extremely important because it relieves the resources for allocating them elsewhere. A project is ready for closing when its activities are performed and relevant follow-on activities have been defined and its result is approved with acceptable quality.

People tend to be forever hopeful that the next software project will proceed infinitely smoother than the last. But history repeats itself in war, economics, love, and SW development projects. (Whitten 1995) Lessons can be learned from each and every project, even if the project is a failure. The main issues during the project should be written down to prevent making the same mistakes again. These issues include the information about the main history of the product with its problems and achievements for archiving such information for future use. (PMI 2001, p. 109) Many companies do not document lessons learned because employees are reluctant to sign their names to documents that indicate they made mistakes. Still, today's business world is emphasizing on documenting lessons learned. It is considered to be an effective way for avoiding re-iterates the same mistakes. (Kerzner 2003)

The road ahead for SW engineering is driven by SW technologies. Reuse and component-based software engineering offer the best opportunity for order of magnitude improvements in system quality and time to market. In fact, as time passes, the SW business may begin to look very much like the HW business today. There may be vendors that build discrete devices, other vendors that build system components and system integrators that provide solutions for the end-user. (Pressman 2001)

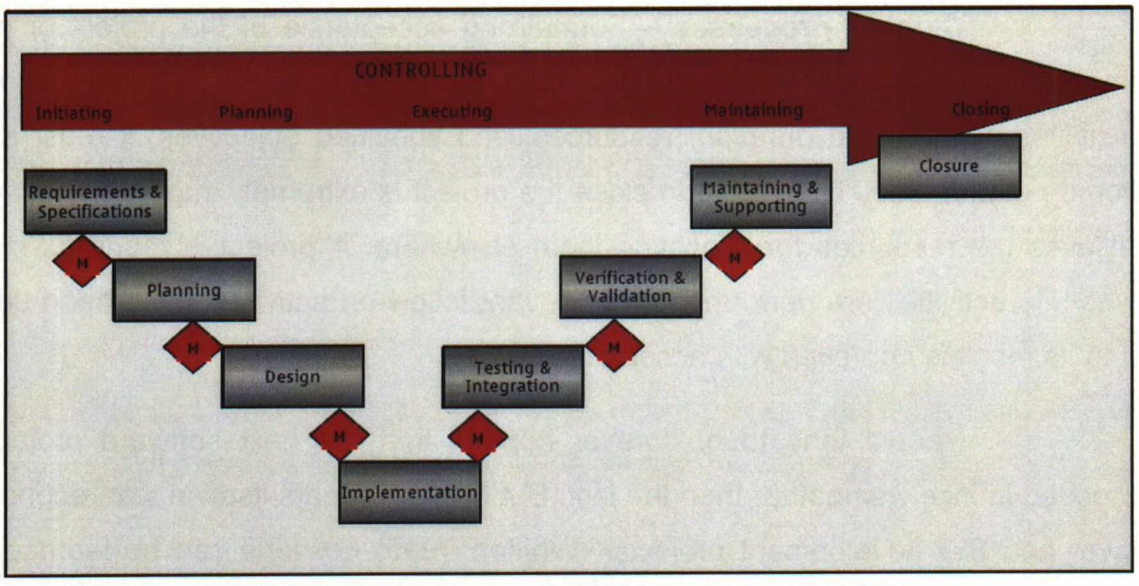

Figure 4. Project and process phases operating in a same time frame (adapted from Armstrong 2001, p. 25; Rahikainen 1997, p. 28 and Kerzner 2003, p. 73).

2.4 Project Management in Distributed Environment

More and more companies today are relying on virtual teams to get things done. There are a number of benefits from choosing a virtual team for the project. First of all, the possibility of having the desired competence at the right time inside the project. Then, people can work from anywhere at anytime, many physical handicaps are not a problem, expenses associated with travel, lodging, parking, and leasing or owning a building may be reduced and sometimes eliminated.

Reasons for virtual teams center around the differences in time and space for team members.

- Team members may not be physically collocated.
- It may not be practical to travel to meet face-to-face.
- Team members may work different shifts

Through virtual team, it is addressed the problem of managing large software projects to improve the economics of their planning and execution. This is a

critical problem that can radically benefit from the rapid emergence of the Internet and other global networks and their role as a global dis-intermediated service market place. In this chapter, we will only refer to the virtual teams located at different sites from the same organization.

Current project planning faces logistical problems such as identifying and assembling skilled teams of people, the timely procurement of products and services, and the optimal decomposition of a project into pieces.

The distributed development of a project involves first of all well defined knowledge of the project requirements and how these can be fully covered and accomplished by dividing the work between sides. The establishment of this should be prepared in advance, letting way as little as possible for unpredicted events leading to miscommunications and lack of skills for all the parties.

2.4.1 Managing the Project

The project owner plays a key role in assembling the virtual team. A typical scenario would be:

1. The project management team makes a decision of using teams located at different sites

2. Project owner announces the software project (project description) and notifies "interested parties" via focused email, newsgroups, or mailing list broadcasts; directing them to an Internet *site* for the project.

3. The parties involved visit the project site and gather information. They set together, under project manager's supervision, the way they will communicate and interact throughout the whole project.

Once the virtual team is assembled, the next step is to design and instantiate a workflow that models the integrated, multi-organizational project. A typical scenario involves:

1. Project owner defines and/or refines the software development workflow. The definition involves specifying:

- The **tasks** and their sequences: such as *design, prototype, implement, unit test, build, integration test,* etc.

- **Roles:** *owner, subcontractors, designers, programmers*

- **Tools** associated with tasks & roles: *project management tools, configuration management tools, programming environments, testing environments,* etc.

2. A software development process for managing the team is instantiated on an Internet-enabled server.

3. The workflow ensures the logical progression of work across the multiple participants and can be monitored and visualized using tools, both by the owner and participants.

Key technical issues are:

1. The Internet will serve as the networking backbone for the transfer of work items, project parts, and results between participant organizations.

2. The workflow model used must be rich & flexible enough to support the concept of multiple participant organizations, where each organization selectively exposes its resources and internals to its project partners.

3. When a part of the project is assigned to a participant organization, it is up to the latter to assign its own resources to the project steps. The project owner does not have any control over specific assignments as long as they conform to the project constraints.

4. It may not be feasible to expect multiple participant organizations to use the same set of tools and environments. Each participant may utilize local tools and environments.

2.4.2 Realities and Challenges of Virtual Projects

Critical to the success of team projects is unstructured & semi-structured **communication**. A well-established infrastructure for team communication is

absolutely essential to the success of a virtual team. The social hurdles of enabling virtual teams may very well outnumber the technical challenges.

Virtual teams are supported by both hardware and software. General hardware requirements include telephones, PCs, modems or equivalent, and communication links such as the public switched network (telephone system) and local area networks. Software requirements include groupware products such as electronic mail, meeting facilitation software, electronic whiteboards, and group time management systems.

Security is an important consideration for virtual teams especially if transmitting sensitive information over a network. Norton Internet Security is a readily available software package. Many email and conferencing products support various types of encryption or other security methods and it may be prudent to check with your suppliers about this

Developing a software involves from the very beginning a **consistent architecture**. The division of work and the channels of communicate each part's ongoing work and results should be well defined. Usually, each site develops different components, isolated from each other, facing discrepancies between their parts. Therefore, well-defined requirements and specifications for each component and good knowledge about overall architecture of the product are very important. The differentiation of the processes followed by the both sides and lack of information about them can be a limitation in the product's development as well. To be able to successfully run a multi-site project, a clear understanding of the architecture and the inter-correlations between modules is important. Based on this, the plan and processes can be designed better and the share of work wisely divided. Based on these observations, we can generally conclude that: "**Consistency from the beginning in architecture, plans, and processes is important as coordination mechanism in multi-site projects**"

The informal communication between members of the project is very important in developing ideas and preventing miscommunications. It is easier for the developers located at the same site to find out which is the best person to contact in a specific situation. They will not spend time writing e-mails and waiting for the replies, the information they are looking for can be easily asked around. Obviously,

the same information will be harder to find from a different site. That is why very important is to know the team members in advance and each member's responsibilities. This way, the communication would be easier and faster. For multi-site projects, we might say, "**The communication between project members from different sites is facilitated by sharing information about each person's role in the project**"

The idea mentioned above is strictly correlated with the next one, "**The most obvious obstacle to communicating across sites is the inability to share the same environment and to see what is happening at the other site**". Difficulties that may arise in communication between different sites can have as basis the lack of knowledge of each part's environment and no coordination in setting the same environment. If all sites would work following the same procedures and systems, some of the problems raised by miscommunication could be overtaken. Also, the differences in each site's culture are very important in communicating and solving the problems. That is why, the human liaisons between sides should be created preferably at the beginning of the project, by permitting the members to make contact to the other site at its location and get to know better the people and their way of working. The communication can be eased for example by videoconferences and net meetings. One important issue to be dealt with in multi-site projects is the barriers of language. English is the current language used in multinational companies and therefore some measures for a good communication between an English and non-English site should be considered at the beginning and also some ways not to let the language be a showstopper for the e.g. project timelines. Taking all these considerations into account, we conclude that "**The inability to share the same environment, as an obstacle of communication between sites can be overtaken by an earliest communication and designing common procedures**". This idea also contains the importance of common documentation and its availability for all the parties involved.

In the context of virtual teams, **the leader position** is even more important. His first key attribute should be to coordinate the teams situated in multi-site environment, paying good care to the people.

Unlike rational organizational structures of the past, teams rely on employee empowerment rather than management control and direction. Team organizations have created work structures that are more democratic and flexible with a common mission of sharing responsibility for results and decisions between management and workers. The ideal team is characterized by a global rather than departmental focus. Problems are controlled at the source rather than by a separate policy function. Information tends to go to employees and there is more of an emphasis on work and home life balance as opposed to long hours. Continuous improvement is highly valued. Instead of promoting employees with highly specialized skills, team-based operations focus on creating flexible, cross-trained and multi-skilled team members. Self-managing teams are said to be the key to leaner and more flexible organizations capable of adjusting rapidly to changes in the environment and technology (Fisher and Fisher, 1998). Presently, people work across internal organizational boundaries such as specialized functions and departments as well as external organizational boundaries. Virtual teams explore a new type of boundary-crossing organization utilizing technology and information.

Fisher and Fisher (1998) define teams as no authoritarian organizational structures commonly used for shared responsibility and employee empowerment. With the advent of so many communication technologies, organizations are seizing the opportunities to work together apart. Like traditional types of teams, virtual teams engage a group of individuals to work independently towards a common goal. Unlike conventional teams, a virtual team works across time, space and organizational boundaries with links strengthened by webs of communication technologies (Lipnack and Stamps, 1997).

The success of a project involving a virtual team depends a great deal of the manager's capabilities to hold the teams together and create the links, the interactions and channels that weave the fabric of the team. The nature and variety of these links are the most distinguishing factor between virtual and traditional teams. The table below displays the principles that provide an integrated framework for understanding and working in virtual teams as Lipnack and Stamps see them.

The inputs needed to develop virtual teams include independent members, cooperative goals, and multiple media. Throughout the development

process, the members share leadership and engage in interdependent tasks, which involve boundary-crossing interactions. The generated outputs include integrated levels of organizations, concrete results and trusting relationships.

Virtual Team System of Principles

	Inputs	**Processes**	**Produced Outputs**
People	Independent Members	Shared Leadership	Integrated Levels
Purpose	Cooperative Goals	Interdependent Tasks	Concrete Results
Links	Multiple media	Boundary-crossing interactions	Trusting Relationships

Table 2. "Virtual Teams", Lipnack and Stamps, 1997

Virtual teams are composed of individual members with certain areas of expertise. Three elements of virtual teams allow them to achieve their purpose: cooperative goals, interdependent tasks and concrete results. Virtual teams rely upon a clear purpose because of their cross-boundary work. Cooperative goals define the outputs desired, while interdependent tasks connect those desired outcomes to those achieved. When a team has completed its process, it expresses its purpose as concrete results.

Links are what give virtual teams their distinction from in-the-same-place organizations. Multiple media (wires, phones, computers, etc.) are the channels by which the members make the physical connection. These connections allow communication and boundary-crossing interaction that make virtual teams truly different. Through interactions, people develop trusting relationships in their patterns

of behavior that persist and feed back into subsequent interactions. While it can be argued that trusting relationships are needed by all teams, they are even more important to virtual teams because of a lack of face-to-face time.

The team leader is the principal player and the success of the entire team depends on him. Of course, no matter how good the manager is, the environment puts also its mark upon the project. The ability to deal with different boundaries created by differences in culture, way of working and to create the environment for an active collaboration between the team members makes the success or failure of the distance projects.

The interactions and distributed environment make the difference in the activities of the traditional teams and virtual ones. The activities that each individual performs in a virtual team depend on the cooperation and influences of different elements upon his performance/efficiency/motivation. After all, the success of a project (virtual or not) relies on the team members, their skills, motivation, commitment and teamwork. Personal relations are very important between team members.

The managers had been working from the common misconception that all it is needed to do to realize a virtual project or even become a virtual company is to supply the employees with the right electronic tools, such as E-mail and remote databases. Nothing could be further from the truth. Virtual operations come with their own set of management challenges, and managers must be skillful enough to recognize those problems and solve them.

Business is, at its heart, a highly personal activity, and to date we don't have electronic tools that can fully replace the richness of face-to-face contact. Whenever one substitutes electronic tools for a physical work space, one loses the synergy that can come only from daily informal contact, and you risk alienating workers from one another and from the company's goals. Productivity is likely to droop if managers are not alert to potential problems like reduced informal contact, improper workspaces, and increased friction between remote and onsite workers.

To lead a virtual team, project managers should be also in the position to detect and solve different issues related with cultural differences, different

management styles and ways of communicating skills. Attention should be paid to improving delegation and empowerment, increasing tolerance and understanding, knowledge and awareness of common standards.

Information and progress should be shared (knowledge work), new ideas can be exchanged (work with people) and potential conflicts can be averted. In many cases, this type of communication can be achieved through the use of various media, such as telephone, video conferencing, personal letters and e-mail.

However, it will also be necessary for the project manager or board to visit the virtual team members in order to provide additional briefings and familiarity. Visits to virtual team members will also provide opportunities for informal discussion on topics not appropriate for electronic communication. Regular contact and communications by the project management and the team can help maintain the team spirit in a number of ways:

- Encouraging familiarity and trust between members;
- Motivating team members and gaining respect;
- Providing familiarization with management styles;
- Resolving cultural and work related issues;
- Team members identify with the project and feel part of the overall work.

3. RISK MANAGEMENT IN SOFTWARE ENGINEERING - LITERATURE REVIEW

In this chapter we will take a closer look on the existing documentation about risk management. The literature is nowadays rich in books and dissertations on this subject. It is very difficult to make a comprehensive analysis and assessment on the quality and practicality of these documents. Therefore, we will present an overview of the most recognized theories and we will choose for circumstantial discussion the ones, which present importance for this paper.

Risks in software development were not addressed in any detail until late 1980's, when Boehm (Boehm 1988; Boehm 1989) proposed and synthesized more detailed approaches for risk management. His work was complemented by Charette (Charette 1989) and software engineering risk management is now an established area within software engineering community. The Software Engineering Institute and annual software risk management conferences acted as the main forum to share experiences and results between practitioners and researchers (SEI 1993; SEI 1994; SEI 1995; SEI 1997).

Some advances in software risk management have produced well-documented approaches for risk management (Karolak 1996; Michaels 1996; Pandelios et al. 1996; Hefner 1994), several categories of risks have been proposed (Chittister & Haimes 1993; Carr et al. 1993; Laitinen et al. 1993; Boehm 1989), quantitative approaches for risk management have been proposed and used (Bowers 1994; Fairley 1994; Berny & Townsend 1993), and there are several software tools available for risk management. Furthermore, most commonly used software engineering standards require some form of risk management to take place, although they do not provide detailed requirements on risk management. (DoD 1988; ESA 1991; IEEE 1992; IEEE 1987; ISO 1994; ISO 1991b; Singh 1991; IEEE 1992; Paulk et al. 1993a; Koch 1993).

Industrial reports on software risk management are relatively rare with some notable exceptions (Boehm 1991; Chittister et al. 1992; Eslinger et al. 1993; Fairley 1994; Gemmer & Koch 1994; Hefner 1994; Laitinen et al. 1993; Meyers & Trbovich 1993; Morin 1993; Williamson 1994; Conrow & Shishido 1997). None of these reports has been able to provide concrete, quantifiable data about the benefits of risk management methods, although they do provide indications that some benefits exist.

Barry Boehm's work has been the main foundation for most of the risk management work in software engineering (Boehm 1981; Boehm 1987; Boehm 1988; Boehm 1989; Boehm 1991; Boehm 1992). His main contributions have been in establishing the risk management as an important field of study in software management, introduction of some key measures for risk, and synthesizing a set of techniques into a single framework for risk management.

Boehm's spiral life cycle model was the first life cycle model to incorporate risk management explicitly in it (Boehm 1988) and many recent papers on software life cycles have incorporated similar notions of risk in them.

Risk Management	**Risk Assessment**	Risk Identification	Checklists Decision-driver analysis Assumption analysis Decomposition Brainstorming
		Risk analysis	Decision analysis Network analysis Cost models Quality factor analysis Performance analysis
		Risk prioritization	Risk exposure Risk reduction leverage Compound reduction

	Risk Control	Risk management Planning	Buying information Risk avoidance Risk transfer Risk reduction Risk element planning Risk plan integration
		Risk resolution	Prototypes Simulations Benchmarks Analyses Staffing
		Risk monitoring	Milestone tracking Top 10 tracking Risk reassessment Corrective action

Table 3. Boehm's risk management model

A major contribution of Boehm was the consolidation of some main techniques for risk management into a single framework. He divided risk management into two main aspects, *risk assessment,* and *risk control.* These were further divided into steps that were supported by a set of techniques. Table 1 presents Boehm's risk management model. The right-most column presents the techniques that can be used to support each step.

SEI's Software Risk Evaluation method has been developed to support systematic risk evaluation (Sisti & Joseph 1994). The method has been also extended to support teams in risk management (Pandelios 1996) and SEI has started collecting their assessment results into a database for further analysis of identified risks (Monarch et al. 1996). SEI's method is structured around a set of continuous tasks that guide the risk management process:

- *Identify*: The method relies on SEI's risk taxonomy to identify potential risk areas (see Table 2).

- *Analyze*: Transforming data from the identified risks into decision-making information. The SRE approach recommends using two alternative, table-based approaches for ranking risks.

- *Plan*: Plan risk mitigation, i.e., define and rank actions to mitigate risks, prioritize actions, and integrating them into an executable risk management plan.

- *Track*: Monitoring the status of risks and their mitigation actions along with the use of metrics and triggering events.

- *Control*: Correcting the deviations from planned risk mitigation actions by using existing program or project management control functions.

A. Product Engineering	B. Development Environment	C. Program Constraints
1. Requirements	**1. Development Process**	**1. Resources**
a. Stability	a. Formality	a. Schedule
b. Completeness	b. Suitability	b. Staff
c. Clarity	c. Process Control	c. Budget
d. Validity	d. Familiarity	d. Facilities
e. Feasibility	e. Product Control	**2. Contract**
f. Precedent	**2. Development System**	a. Type of contract
g. Scale	a. Capacity	b. Restrictions
2. Design	b. Suitability	c. Dependencies
a. Functionality	c. Usability	**3.Program Interfaces**
b. Difficulty	d. Familiarity	a. Customer
c. Interfaces	e. Reliability	b. Associate Contractors
d. Performance	f. System Support	c. Subcontractors
e. Testability	g. Deliverability	d. Prime Contractor

f. Hardware Constraints g. Non-Developmental Software **3. Code and Unit Test** a. Feasibility b. Testing c. Coding/Implementation **4. Integration and Test** a. Environment b. Product Integration c. System Integration **5. Engineering Specialties** a. Maintainability b. Reliability c. Safety d. Security e. Human Factors f. Specifications	**3. Management Process** a. Planning b. Project Organization c. Management Experience d. Program Interfaces **4. Management Methods** a. Monitoring b. Personnel Management c. Quality Assurance d. Configuration Management **5. Work Environment** a. Quality Attitude b. Cooperation c. Communication d. Morale	e. Corporate Management f. Vendors g. Politics

Table 4. SEI's risk taxonomy

The above steps are visually represented in Figure 5.

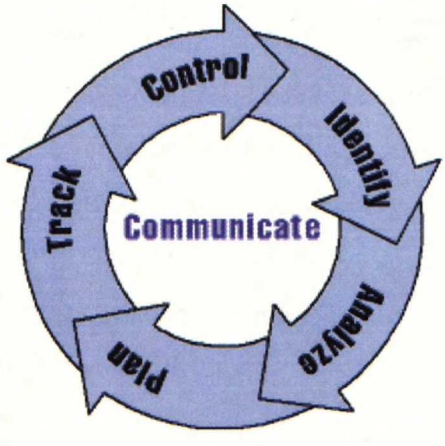

Figure 5: SEI's risk management cycle

A central element in SEI's approach is the risk taxonomy and associated questionnaire. The taxonomy is presented in Table 4. The taxonomy is covered with

a questionnaire that, through its 194 questions, covers all areas listed in Table 4. For more information about the questionnaire, please see the reference (Sisti & Joseph 1994).

The SEI method documents risks using risk statements. Risk statements document risks in condition – consequence pairs. The condition attribute contains a sentence describing the situation and the consequence attribute describes the outcome of the current condition if a risk occurs. The risk statements can be used visually, as shown in Figure 5, or on textual basis, as presented below:

> (Given the condition that)
>
> We must use Linux and we have little experience in it;
>
> (Then there is a concern that)
>
> The implementation phase may last longer than planned.

The SEI's method is well defined and requires broad participation from the organization that is using it. A typical risk management cycle lasts two to four months (Sisti & Joseph 1994). Given its higher costs it seems that the SEI method is suited for assessing program level in the beginning of a large program. It provides training and understanding on risk management issues while identifying risk areas. The SEI Continuous Risk Management Guidebook (Dorofee et al. 1996) is one of the most comprehensive collections of practical techniques that can be used in various steps during risk analysis. However, while being a practical and easy to use, the guidebook contains hardly any theoretical introduction to risk management and it does not discuss many key limitations and biases associated with the techniques it presents.

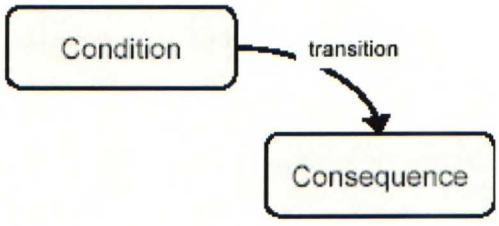

Figure 6: Visual format for the SEI risk statements

Hall has proposed a five-level capability maturity model for risk management, the RM-CMM (Hall 1995). The model contains a set of factors that are used to assess the maturity of a risk management system. The factors used in Hall's model are presented in Table 5.

Dimensions		Risk Management Evolution Framework				
		1- Problem	2 - Mitigation	3 - Prevention	4 - Anticipation	5 - Opportunity
Process	Identify	Not seen as positive	Risks are assessed	Risks are volunteered	Risks are sought out	Chances to do better
	Analyze	None	Prioritize risks	Analyze source of risk	Quantitative values used	ROI is calculated
	Plan	None	Action plan is discussed	Action plan is documented	Action plan is executed	Action plan is revised
	Track	None	Monitor critical risks	Monitor all risks	Monitor triggering events	Correct deviations
	Control	None	Discussions increase awareness of what could be improved	Written evaluations document what could be improved	Written evaluations are analyzed and documented as lessons learned	All feedback is tied to improve the process
Infrastructure	Policy	No written standards	Report risks at reviews	Commitment to process	Commitment to metrics	Reward for innovation
	Communi-cate	Lack of communicatio n regarding risks	Risks gathered from lower levels and not communicated to higher levels	Communicate risks within the program team	Between the program team and the customer	Between the program team, customer and the end-user
	Commitment	Upper management	Quality assurance	Management	Employees	Customer and end-user
	Resources	None	Minimal schedule allocation	Minimal schedule and budget are allocated	Sufficient schedule, budget and some resources	Optimal schedule, budget and resources are allocated
	Training	No training	Basic risk concepts	Risk management process	How to quantify risks	How to manage risks
Implementation	Participants	Program manager	Program manager and key technical staff	Program team with a single risk champion	Program team and customer, and a few risk champions	Program team, customer and end-user, with many risk champions
	Procedures	Ad hoc	Verbally stated	Documented	Updated milestones	Living document
	Methods	Ad hoc	Risk surveys	Risk taxonomy	Risk management form	Risk metrics graphs
	Tools	Ad hoc	Top 10 risk list	Risk database	Technical performance measures	Automated risk analysis
	Metrics	None	Defined	Collected	Analyzed	Reported

Table 5. Hall's model

Although Hall presented some survey data to support the model. It perhaps should not be seen as a normative maturity model but a framework for identifying risk management issues. Hall has also developed a comprehensive risk management approach that includes risk management process definition, description of the risk management infrastructure, and guidelines for implementing risk management in practice (Hall 1998). Hall's approach also incorporates goal setting, project planning, execution, measurement, improvement, and discovery of new information into a conceptual framework for project execution and improvement. This

"six-discipline model" can be considered an improvement paradigm that consolidates some aspects of risk management into it.

Karolak has developed a risk management approach that is based on identifying a set of high-level risk categories, called risk elements by Karolak, associating risk factors to them, and, again, associating specific risk metrics, or questions, to these factors (Karolak 1996).

Questions are answered by project representatives, answers are converted to numerical values, and the network of answers and their weights are used to calculate risk factor values, using probability tree calculations. Karolak's model gives quantified estimates of project's risks along the risk factor categories. In addition, the questions in the model can be used as checklists to identify specific risks. There are several other proposed models that use the same principle of trying to use situational factors (Foo & Muruganantham 2000; Roy & Woodings 2000; Deutsch 1991; Madachy 1997; Groth 1992), software component characteristics (Briand et al. 1993b; Briand et al. 1993a; Madachy 1997; Madachy 1997), or software architecture characteristics (Weyuker 1999) to predict project risks and using to predict projects risks.

The influence of the motivation, attitudes, and organizational context to risk management has been recognized by several authors. Gemmer studied practices within Rockwell and observed that current existing cultural rules acted as disincentives for proactive risk management (Gemmer 1997) and called for effective communication to change the climate to be more accepting to risk management. Hall has also recognized the importance of providing motivation and skills for risk management, as well as involving people form all levels in risk management (Hall 1998).

The traditional engineering fields have also addressed risk management over the past decades (Michaels 1996; Petroski 1985; Ricci et al. 1981; Waller & Covello 1984; Wang & Roush 2000). In particular, the Failure Modes and Effects Analysis (FMEA) approach is commonly used during the design of hardware products to identify potential design flaws by reviewing specifications and designs, and to estimate their impacts (Stamatis 1995). FMEA has been extended to include the evaluation of criticality and risk (FMECA), i.e., the method results in identification

of most severe failure modes. Even though software is often more complex and its components not as clearly decomposable, the FMEA can be used to analyze risks associate to software operation. However, the method is not equally well suited to evaluating software development risks as software development project and organization cannot necessarily be decomposed in a way that it would help analyze main risks.

In summary, the software risk management has been an active research topic during the past 15 years, resulting in a comprehensive portfolio of approaches. However, as we will discuss in the next chapter, many of the proposed approaches have problems associated with them and these limitations are rarely discussed.

3.1 Limitations of Current Approaches

As the previous discussion shows, there is no shortage of proposed approaches for risk management. It is perhaps surprising that many existing approaches have various limitations that are often not acknowledged or addressed by authors or by practitioners. The reason for this may be that risk management in software projects always contains a dilemma: one expects reliable results with little effort and investment. After all, a project's goal is to deliver products, not to spend all of its time on pondering hypothetical problems.

Nevertheless, failure to account for the limitations in the risk management approach used may result in serious bias in risk management results.

In this chapter, we highlight some of the main limitations that affect the applicability of many risk management approaches. Sometimes these limitations may have little practical relevance and they do not necessarily indicate that a risk management method does not work in practice. However, most of them have a high potential for creating bias in risk analysis so we recommend that any risk management program should take a conservative position and "prove" that the limitations are not serious in their situation.

Many risk management approaches address a limited number of goals, such as schedule, cost and product quality (Gemmer & Koch 1994; McCaugherty 1996; Sisti & Joseph 1994).

There are many cases where projects actually have important other goals that eventually affect projects success, such as impact on reputation, ability to reuse projects results, compliance with constraints set to the project, need to maintain compatibility with other systems, and process conformance requirements. In fact, project's goals can rarely be truthfully expressed in two or three goals. If a risk management approach limits its risk identification and loss evaluation approaches to too few goals, some risks may be ignored or ranked lower than they should.

Few risk management approaches explicitly recognize the different expectations different project participants, stakeholders, have on the project and its goals. Sometimes the customer and other stakeholders are involved in the risk evaluation process (Sisti & Joseph 1994), but the involvement of such parties does not necessarily guarantee that their interests are supported in risk analysis phase. Even if other stakeholders are involved in the analysis, a joint, consensus-based ranking of goals may not be the most effective mechanism to deal with these stakeholder perspectives: it may increase communication overhead and sometimes politics prevent open discussion of critical issues when different stakeholders are present.

Many organizations have attempted to streamline their risk identification processes by developing risk checklists or taxonomies (Bezirkan & Mulazzani 1994; Boehm 1989; Carr et al. 1993; Jones 1994; Rook & Cowderoy 1993; Speaker 1993). These can be helpful tools in making sure that all previously identified categories of risk are covered. However, such taxonomies may also increase the tendency of participants to focus on the issues covered by the checklist and limit their ability to use their independent judgment to identify risks outside the checklist. The results of such taxonomy or checklist based risk assessments are sensitive to the appropriateness of the taxonomy used for the project and situation. If the taxonomy does not cover the "right" risks in a situation, the results are likely to be wrong.

Another potential problem with taxonomies is that they inherently contain trade-offs between coverage, detail, and user fatigue. A taxonomy that has broad

coverage and is detailed may result in user fatigue: they tend to become less alert towards the end of the taxonomy list, possibly failing to recognize some risks.

Quantification and ranking of risks is a widely recognized challenge in risk management (Boehm 1991; Charette 1989; Friedman 1993). Normally risk ranking is based on estimating probabilities and losses. Probability estimates can be based on three main approaches: historical frequency data, subjective estimates, and estimation tables. All have potential major limitations that are rarely discussed by their proponents. Historical data is rarely used for estimating probabilities, presumably because organizations rarely collect risk occurrence frequency data. This is actually not necessarily a drawback, as historical data's significance in predicting individual events is particularly questionable when situations change (French 1986). As each software projects is unique and technological changes are frequent, relevance of historical data is limited. However, historical frequency data can be used as a sanity check and reference point when the other two estimation methods are used.

Risk estimation tables have been used by many organizations to avoid subjective bias in probability estimates and to reduce the cost of risk analysis (Anon. 1988; Boehm 1991; Charette 1989; Karolak 1996; McCaugherty 1996; Sisti & Joseph 1994; Caplan 1994). These tables typically identify a set of factors – such as maturity of technology, complexity, requirements stability, and experience – and assign some probability score or value based on the "scores" on each factor. This approach has the following potential limitations and assumptions: Tables may produce a list of probabilities whose total may exceed one without any mechanism to account for joint probabilities.

- The same set of factors is used to evaluate all risks. Some risks may be influenced by other factors and the predefined set of factors may be a poor predictor of probability for such factors.

- The factors use same weights for all situations and risk items. The allocation of weights to factors is a subjective process and few, if any, table-based probability estimation approaches give details how the weights have been derived.

Even if the weights are assumed to be representative in the general case, they may not be applicable in all situations.

- Scaling of scoring values for each factor is critical. Marginal increase of a value for a factor and its impact on probability should remain constant for all factors within a factor's value range. This is especially true if factor values are used in mathematical calculations. In other words, one should be able to assume that factor value scales are distance or ratio scale metrics

- Consolidation of factors should be based on the allowable mathematical operations, given the type of metrics used for factors. If factor values are not represented in absolute, ratio or distance scale metrics, they cannot be added or multiplied.

The above limitations are rarely addressed by table-based risk probability estimation approaches. The use of such tables is likely to lead to a consistent and low-cost, but unreliable and inaccurate risk estimation process.

Subjective probability estimates reflect a person's belief in the likelihood of a risk occurring (French 1986). Despite the subjectivity of such a definition of probability, there is a growing amount of research in dealing with such estimates (Kahneman et al. 1982; Tversky & Kahneman 1974; Kahneman & Tversky 1973). Subjective probability estimates, especially when done by individuals with access to past history data and good understanding of the domain, are perhaps the most reliable mechanism to estimate probabilities of future events.

Human experts may intuitively be able to process domain information to yield best available probability estimates. In order to compensate for individual bias, such estimates should be collected from several individuals and results discussed to consolidate differences.

Several different approaches have been proposed for the estimation of losses. The most obvious method is subjective estimates; perhaps the most widely used approach. In addition, Boehm proposed the use of cost models, network analysis, and quality factor analysis (Boehm 1989).

When a risk affects more than one valuable characteristic (a goal) in a project, the ranking of losses easily becomes non-trivial. Table-based estimation approaches have been used for loss estimation by many of the same approaches that use table-based probability estimation (Anon. 1988; Boehm 1991; Charette 1989; Karolak 1996; McCaugherty 1996; Sisti & Joseph 1994) and the same, often serious, limitations apply to such estimates. Although the decision analysis field has studied multiple criteria decision-making problems extensively (Saaty 1982; French 1989), methods from that field have not been applied in software engineering risks management, despite their obvious potential.

However, most of the approaches based on such tables (Sisti & Joseph 1994; Speaker 1993) use them inefficiently and fail to identify rankings within classes, resulting in unnecessary lack of precision, or do not include the rationale for the table priorities used (Greer et al. 1999; Newland et al. 1997).

Few risk management approaches give an accurate definition of risk. They mainly refer to risk as a "possibility of loss". In practice this definition leaves open several alternative interpretations of risk, such as the actual loss that would result if the risk occurs (Anon. 1992), a factor or element that is associated with a threat (Anon. 1992), probability of a risk occurring, or a person that contributes to the possibility of loss (Anon.1995a).

In summary, the software engineering risk management practice is using several methods that have potentially serious limitations of biases and the literature in the field rarely addresses these problems. With few exceptions (Conrow 2000), literature does not contain critical discussions of these limitations. We are afraid that as a result, practitioners are largely unaware of these limitations and continue to use such methods in critical software projects. The purpose of our paper is to identify and clarify how the industry practices versus risk management can be improved, as we believe that most organizations perform little systematic risk management in their projects and even use biased or incorrect methods to analyze their risks.

3.2 Risk and Standards

The importance of risk management has also been recognized by some software engineering and quality standards. While these standards may not be the driving force in making risk management more common in industry, they do represent a gradually improving level of more systematic risk management in industry.

The IEEE Standard 1074 for Developing Software Life Cycle Processes (IEEE 1992) considers risk analysis a mandatory activity, requiring that "risk management is performed throughout the project's life cycle" and that "technical, economic, operational support, and schedule risks are identified and analyzed". However, the standard only gives recommendations on how risk management is carried out, it may include "modeling, simulation, prototyping, independent reviews and audits". The IEEE 1074 standard is a widely accepted and used standard in the U.S.

The IEEE standard 1058.1-1987 for project management plans describes the requirements for project management activity (IEEE 1987). One of the requirements is risk management, which is also an explicit section in the recommended project plan. The standard requires that risks are "identified and assessed" and "the mechanisms for tracking the various risk factors and contingency plans" are prescribed.

The U.S. DoD standard 2167A (DoD 1988), that describes the required software development processes of DoD contractors, states that contractors must "document and implement plans for risk management" and that the contractor shall "identify, analyze, prioritize, and monitor areas of the software development project that involve potential technical, cost, or schedule risk". The standard does not provide any further requirements.

The ISO 9000-3 guideline (ISO 1991b) for applying ISO 9001 standard (ISO 1987) to software does not address risk explicitly. However, the ISO 9000-3 does require that personnel have "freedom and authority to initiate action to prevent the occurrence of product nonconformity". Furthermore, it also requires that during contract review "possible contingencies or risks are identified". In summary, ISO

9000-3 only presents minimal and very general requirements for risk management and it cannot be said that ISO 9000-3 would support risk management. However, risk management would clearly contribute to the overall objectives of ISO 9000-3.

The Capability Maturity Model (CMM) of the Software Engineering Institute (SEI) requires risk management on level 2 (Paulk et al. 1993a). The process area of *project planning* has an activity (Activity 13) that requires that "the software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented". The same activity also requires that "risks are analyzed and prioritized based on their potential impact to the project" and that "contingencies for risks are identified". The key process area of *software project tracking and oversight* also requires that risks are tracked during the project and that "high-risk areas are reviewed with the project manager on a regular basis". Some other key process areas, such as *software quality assurance* and *software quality management* implicitly require some risk management activities to take place.

The IEC/ISO standard 15504, also known as the SPICE model (Dorling 1993), defines a framework for the assessment of software processes (ISO 1998c). Its reference model (ISO 1998b) defines requirements for the risk management process. It states that organizations should define the scope and strategies for risk management, identify and analyze risks, define metrics for risks, and take action to reduce risks.

The ISO draft standard Information Technology Software Life-Cycle Process (ISO 1991a) also addresses risk, although only briefly: projects should manage technical, cost and schedule risks.

The British standard 6079 define a process for identifying, assessing, and controlling risks in projects (Anon.2000b). The standard defines a risk management process, gives guidelines on each step, provides example risk ranking tables, and contains a general checklist for project and business risks. The standard recognizes stakeholders and their impact to risk evaluation.

In summary, the software related standards contain only limited requirements and guidelines for risk management. The requirements are so general that quite simplistic risk management practices satisfy them. It seems that even if the

more recent standards have more detailed requirements on risk management, they are not widely used in the software industry.

3.3 Conclusions on Literature Overview

Risk management is a relatively young but very multi-disciplinary field: as a formal and explicit activity it has been practiced and researched in many fields since the middle of 20th Century. The software engineering risk management will benefit from taking advantage of the contributions in other fields to develop techniques to support risk management. This is particularly important as the current state-of-practice in software development is based on very primitive – and often faulty – techniques. Software development is too important to be controlled by biased or superficial techniques.

The review of relevant literature highlighted several issues and contributions that are used later in this work. First, the requirement for systematic risk management is common in many disciplines, explicit and formal risk management practices need to be in place to ensure sufficient frequency and quality of risk management. Therefore, the methods and procedures presented here are defined with sufficient detail and rigor that they can be applied systematically and consistently.

Second, the role of stakeholders and their perspective on losses should be made more explicit in risk management, as they are the ones that can determine the significance of potential losses of risks.

Finally, risk management practice and understanding must be continually improved, both from the perspective of software industry, as well as from the perspective of each organization. The software industry is not using state-of-art knowledge and methods in risk management and we need to improve practitioners' awareness of more correct and more effective techniques. Each software development organization should also establish a risk management improvement framework that supports and forces them to learn from their past experiences to improve their understanding of risk and improve their risk management practice.

4. PROPOSED FRAMEWORK FOR STUDY

4.1 Risk Management in Software Projects

This chapter gives a brief introduction on the key practical issues of risk management in software development projects to be addressed in more details in the following parts of this work.

Software projects have the potential to suffer from numerous problems, including missed deadlines, inaccurate budget, unmet specifications, product defects, unforeseen project risks, changing requirements, poor resource planning, poor management etc. These risks have the potential to turn any software project into a disaster, and to ruin a software organization. It is possible to minimise these risks by project planning and management. Developing software by following a defined software engineering process is one method and aid for risk management. (Hutchens et al. 1997)

Ensuring that the plans are realistic is the risk management's area of responsibility. Risk management focuses the project manager's attention on the issues that are likely to cause problems. Risk management considerations can also help the project manager to determine the appropriate sequence of the project activities. The practice of risk management includes two primary steps, risk assessment and risk handling. Risk assessment involves risk identification, risk analysis, and risk prioritisation. Risk handling involves risk management planning, risk management execution, and risk monitoring and control. (Boehm & Ross 1989)

Risk identification produces lists of project-specific items that are likely to compromise the project's conditions. Risk analysis produces assessments of the loss-probability and loss-magnitude for each of the identified risk items, and assessments of combinations of risks that are involved in risk-item interactions. Risk prioritisation produces a prioritised order of the risk items identified and analysed. Risk management planning should then result in two kinds of plans for each risk. One is intended to minimize the probability of occurrence of the risk, and the other is a contingency plan that is needed to minimize the losses if the risk becomes reality

after all. Risk management planning includes the coordination of the individual risk-item plans with each other and with the overall project plan. Risk management execution implements the preventive and contingent actions that were planned. Risk monitoring and control help to indicate possibly materializing risks for being acted upon, and also totally new risks to be addressed in risk management planning. It involves tracking the progress toward resolving risk items and tracking corrective action where appropriate. (Boehm & Ross 1989; Haikala & Märijärvi 1998)

Potential risks that can impact the project must be constantly looked for. Risks can surface internally from the project team members or externally from other project groups, business units, vendors, and management. That is one reason why a good project manager spends much of the time communicating with these entities. The same applies also for potential opportunities that can impact the project in a positive way. (Lientz & Rea 2001) Some people say that the most important rule for risk management comes from Murphy's laws; everything that can go wrong will go wrong. To keep project plans realistic, this may be even too an extreme mindset, but regarding comprehensive risk management, it could be good to keep in mind.

4.2 Key Risk Management Activities

The business environment in which software companies develop their projects does not "allow" business success without a powerful risk management process in place. The framework for risk management analysis and control proposed in this chapter will be taken as basis for the projected cases of this study.

The review of the books and articles in risk and project management fields (presented in the previous chapters) allowed me to deepen the knowledge and comprehend to some extent the profound connection between these two areas. Risk management methods and methodologies should be applied extensively from the earlier phases of the project. The more the risk analysis is delayed or ignored, the more probable the key areas of the project success are affected by uncertainties and failures.

The management team of the project should be aware at the risks that might rise in each of the three areas surrounding a project team:

- Product engineering
- Development environment
- Project/program constraints

Unquestionably, the weight induced by each of these three categories depends very much on the company in question. A project developed in small company is presumably subject to higher risks from product engineering area and less from development environment, since the processes are more flexible and easy to change "on the fly" according to each project characteristics. This is unlikely to happen in a large company, with tens or hundreds of projects, which have to follow company's policies and processes.

However, we approached the framework from the large company perspective for two reasons: one is that the case study proposed takes place in a large Finnish company (Nokia Oyj) and the second one is that it is always easier for whoever might use the findings of this work to choose specific data needed from a multi-layered work than try to adapt small parts to create a bigger picture.

We consider that identifying the above-mentioned areas and factors, as possible risk generators should be the first step in a project. The next step is to apply the risk management methods for each identified risk-related area. Simultaneously, the risks correlated with each phase of development process should be acknowledged. Generally, following phases or risk activities are recognized as being essential for high-quality risk management process.

1. Risk Management Start-up
2. Goal & Stakeholder Review
3. Risk Identification
4. Risk Analysis
5. Risk Control Planning and Control
6. Risk Monitoring

4.2.1 Risk Management Start-Up

The Risk Management Start-Up defines the scope and focus of risk management activities, as well as who is responsible for it and how it should be done. It lays the groundwork for further risk management activities/steps and defines which forum decides on risk management in the respective unit and it also specifies how losses can be categorized.

The scope is to define what areas are included or excluded from risk management. It also defines risk management team's primary focus area and who is the risk owner in the respective field. The main objective is to establish who shall carry out the specified risk management tasks within the process, to whom risks are escalated and who prioritizes risks as well as to define the roles and practices. Three areas should be emphasized:

Frequency: how often risk management practices are performed and situation monitored

How are losses defined

Contact persons for risk management

4.2.2 Goal & Stakeholder Review

In practice this requires finding out various Stakeholders' goals and what interests they have to protect in the case and how those interests are prioritized.

Goals of Stakeholders are to be clearly defined in order to understand what business goals/objectives are to be protected via risk management activities. Goal description must be detailed enough to be able to identify, analyze and compare the most relevant risks for the goal. For example, a goal for sourcing could be stated: "We are to provide high quality (1% bad quality tolerated) components at the right time (0,5 day tolerance) in a cost-efficient way (material costs x% of the end product) to production line".

Stakeholder is an organization, person or team who can affect or can be affected by our activities or results. Examples stakeholders include the customer, other business units, employees, account/customer teams, product programs, other functions, such as marketing, production line etc. Stakeholders/goals can be identified for example in a risk identifying workshop/brainstorming session or in a separate workshop.

Each key Stakeholder must be named on a sufficient level and his/her goal must be described clearly. Usually it is better and time is saved if only the most relevant Stakeholders are chosen for analysis, i.e., those that most likely shall have clear interest on the result of the work being performed. Various Stakeholders may have different priority for goals even though the same activity is in focus and, therefore, priorities must be known (e.g., in a turnkey project stakeholders are: Customer, Nokia customer account teams, Project team etc. Priority of a time schedule might be different for the customer and project team).

A goal typically takes both (business/strategic etc.) objectives and constraints into consideration. Goal shall be stated in consensus with the unit performing work, e.g. customer account team's goal is to ensure sales of a customer project, with which customer is satisfied and which meets Nokia's profitability expectations, both on short and long term (time schedule, requirement specifications, credit questions, quality aspect, retrofits, etc.) Simultaneously, product program may have somewhat different goal description or various elements of goal have different priority (time schedule, quality, costs etc).

When each stakeholders goals and their relative priorities are known, risk management can be planned in accordance to prioritized objectives of all related stakeholders.

4.2.3 Risk Identification

The scope of this phase is to produce an extensive list of all risks that may endanger the achievement of the targeted business objectives.

Extensive risk list could be produced through a Risk Assessment Meeting by experienced/senior personnel and adequate number of persons representing various relevant functions (F&C, Marketing & Sales, Security, R&D designers, IPR etc).

A business owner (e.g., program manager, account manager, project manager etc), or his/her nominee facilitates identification workshop. In these meetings, risks are separated from "problems" (see Appendix 1) and common understanding of risks is formed. Workshop may take 1-5 hours or more, until no more risks are discovered.

Risk List should include all risks regardless on which process or platform area they may "belong" to or on whose responsibility area risk organizationally belongs to. Risk List expands the understanding of risk environment crucial for the success of the project. Risk List as such is only a list, which does not include or require any further analysis at this point. All risks mentioned during the workshop are documented on the list.

Risk List is the basis for further risk management steps like analysis and control. Thus a thorough risk identification session is crucial. As methods and tools for risk identification, can be used: Checklists, Assessment (brainstorming) workshop, drawing risk continuums, drawing risk scenarios.

4.2.4 Risk Analysis

Risk Analysis is done in order to understand, define and prioritize risks. After the analysis phase, most relevant risks are selected for further follow-up (Controlling and Monitoring).

Part of the analysis phase is to walk through the risk list and come to consensus regarding root causes, probability, consequence, priority (see Appendix 1) i.e. true nature of risks in order to be able to plan control activities.

First step is to define risks (deepen the understanding of the risks in original risk list), second is to prioritize them. This is done by identifying the facts and

characteristics (i.e. Risk Root Causes) that may influence/cause risks to occur (i.e. Risk Events), and impacts they will have (i.e. Risk Consequences).

Probability and impact are estimated according to best understanding based on experience. One way is to seek for arguments and data from the history and compare to similar situations happening before. The next step is to impact consequences to effects that risk would have if occurred. Consequences are described or quantified (risk magnitude, see Appendix 1) using the goals defined in earlier steps in the process. The significance of impact is assessed based on how each stakeholder perceives the losses i.e. impacts as represented by the Risk Effect.

Stakeholder ranking of impacts should be based on Utility Theory (see Appendix 1). In practice this means that stakeholders are asked to evaluate which impact is worse i.e. causes them most "pain". Qualitative and quantitative impacts are evaluated as such; monetary dimension is not essential to simulate if not expressly desired. Impact is described by using wording Insignificant, Low Significance, Medium, High, Very High. The analysis phase enables prioritization (probability x impact =>; Risk Magnitude high, medium and low) of risks via these two elements mentioned and understanding the overall utility loss i.e. "pain experienced". However, this analysis will not be part of this study entirely, but only the simplest part of it.

Risk scenario (see Appendix 1) can be produced in the analysis phase. Typically root causes for listed risks are found from various functions, platforms and processes. Root causes can be found out by using risk scenarios e.g., "mind map" type of charts/drawings where risks are presented as continuums. Agree how deep into root cause analysis it is useful to go. Risk scenario is one way to present outcome of the risk analysis. Risk scenario enables linking separate risks into a larger and more complete view on inter dependencies of risks. When root causes and linkages between risks are understood, actions can be targeted in the most efficient way.

4.2.5 Risk Control Planning and Control

Proactive controlling activities are all the means and methods necessary to prevent (or minimize possibility of) a risk from materializing. Controlling actions can also minimize direct monetary or qualitative losses caused if a risk materializes. A formal control plan is important to ensure clear responsibilities, as well as strong and timely actions.

Risk control activities are done beforehand in order to mitigate impacts that risks (if materialized) ultimately would have on targets (e.g., profitability suffers because of being late from markets, credit losses, component failures, delivery delays, brand damage etc). Proactive controlling is key for successful risk management.

Usually risks chosen for controlling are those that in analysis phase are ranked so that probability of occurring and impact is considered too high. This means that either qualitative or quantitative probability of loss would be too big to take as such. Qualitative losses are not necessarily evaluated in monetary terms (e.g., brand damage).

Risk controlling requires agreement on responsibility; agreed/nominated persons are responsible for chosen risks (Risk action owner). Risk action owners are to ensure that controlling activities are planned and documented, implemented and that control activities and their results are regularly followed in a proper forum. Controlling activities may need to be continuous and they can be implemented in and between related processes. Controlling activities can also be planned as separate actions that do not have a permanent place in the process but can be made only for a specific situation.

4.2.6 Risk Monitor

The Risk Monitoring phase includes the follow-up of risk situation and risk controlling action implementation, as well as communication of the risk situation to key stakeholders, especially to management.

Monitoring can be understood as communication, escalation and partially as a decision-making method regarding risks identified and chosen for controlling. Monitoring means continuous ability to raise management attention to new risks as well as share relevant risk information. Monitoring can be performed with specific template or by any other agreed documented format e.g., Excel reports, minutes of the meetings, analysis formats etc.

Monitoring collects together all relevant data regardless on which function or process the risk belongs to and where controlling activities are done in practice. Monitoring is performed through continuous identification and analysis activities of the related teams. Monitoring shall be practiced on certain intervals (e.g. monthly) in order to update understanding on current situation and to be able to make necessary decisions or re-direct course of controlling actions if necessary.

5. RESEARCH METHODOLOGY

5.1 Research Problem and Objectives

The overall goal of this work is to improve the practice of risk management by developing and providing improved methods and insights to support software engineering risk management. We have defined the research problem in this work into two objectives, as listed below:

1. Develop a comprehensive and practical model for risk management in software projects based on literature review.

2. Analyze, evaluate and improve the theoretical framework with the results of the empirical part and create a functional model to be used in software development projects.

Unfortunately, we could not verify afterwards our model in practice to provide information on its feasibility, effectiveness, advantages, and disadvantages and to improve it further more.

5.2 Scope and Contributions

This work focuses on risk management in software development projects or programs. More specifically, we are studying and developing method and techniques for people involved in software projects. The development of the method and its improvement framework included the definition of process, roles and responsibilities, information types, and the templates used in the process.

The results of this work are intended to be applicable to any kind of software development; however, we believe that medium to large organizations and complex projects are more likely to benefit from the results of this research.

The primary contribution of this work is the development of a comprehensive and easy to follow method for software risk management. Its characteristics include a process definition, integration of stakeholders and goals into the risk management process, approaches to risk analysis, the identification of a risk

controlling action taxonomy, definition of various information types and a templates for supporting the risk management process.

5.3 Research Process and Methods

The research process followed consequently the phases described in the diagram below:

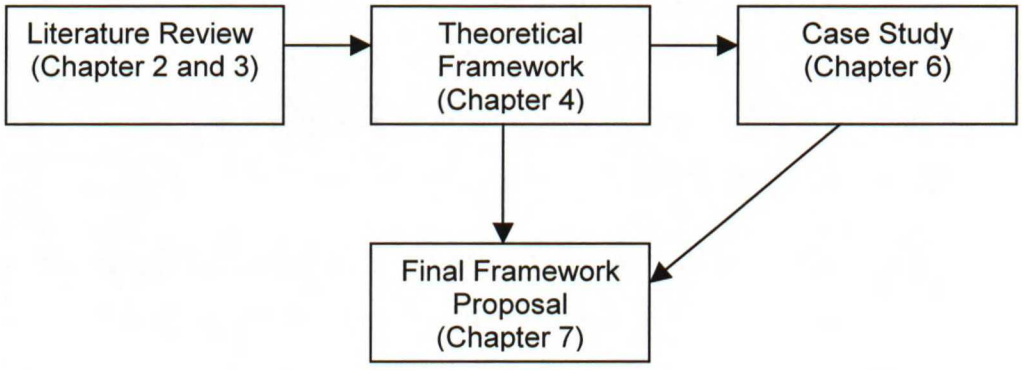

Figure 7. Research steps

The first phase of our research process is the literature review divided in two chapters, with the scope of correlating the risk management procedures in software projects.

The second phase of our research consists of creation of the basis on which we will conduct the empirical part. We have tried to assemble the key ideas of the theoretical part and create a starting point/model proposed for the case study. This is materialized in the integration of risk activities in software projects and identification the key risk activities identified for software projects.

The empirical study describes a software program developed in Nokia Networks. The presentation includes also the overall environment of company as well as the explanation of the program principal participants and activities. The focus is on risks activities, process development and improvement and best practices applied or subject for improvement.

The outcome of this work is a model/framework for risk management workflow, which has as basis the theoretical model of the second phase and it is improved and further built-up with the results of the case study.

As research method, we have chosen the action research for several reasons, presented briefly in Chapter 1.4. Besides them, we considered important that in action research the investigators try to fulfill the needs of their study and, at the same time, generate new knowledge. We have acknowledged the fact that a research environment can be more deeply understood if the researcher is part of that environment. This can be achieved through the researcher becoming an agent of change in the environment, as usually is the case in action research in general. The involvement of the researcher with the environment under study is also believed to foster cooperation and candid information exchange between the researcher and those who are being studied well beyond what can be expected in traditional research approaches. This, in turn, can increase the validity of research findings.

As I was part of the case study presented in this document, I have approached this work by utilizing the action research methods as the best practices for our analysis, hoping to achieve benefits for both *researcher* and *organization*. Our goal was also that t*he knowledge obtained* could be *immediately applied* for next projects under a form of conceptual framework (Baskerville and Wood-Harper, 1996).

In our research, we have followed the five phases of the action research:

1. *Problem Identification,* which can be translated by a thorough integration of risk management activities in the software projects. We have identified the models and adequate principles of risk management to be used in software development projects and key risk activities are recognized as starting point for our empirical research.

2. *Collecting Data* is the second phase of our research, consisting of the overall description of the case study chosen. Collection means the internal environment for the software program (Chapter 5.1), the organization presentation and the driver processes involved. It also includes the description in detail of the software program chosen for study, the team and the inside hierarchy, subprojects and the interdependencies between them.

3. *Organizing Data* phase is the collection of risk management practices developed inside the program and the high level description of the risks mentioned in our case study (Chapter 5.3.1).

4. *Analyzing and Interpreting Data.* The research focuses at this point at analyzing the risks (Chapter 5.3.2) and their interpretation using the key activities model presented in Chapter 4.2 (Chapter 5.3.3).

5. The fifth phase of our research is *Taking Action*. Chapter 6 is our proposal for a risk management model. We have tried to combine information from empirical data analysis with information from the literature review (as they are collected in Chapter 4.2). We evaluated the situation and if/how our framework could improve the outcomes of the management activities. We have investigated also how we can improve the framework by using the practices encountered in this project.

6. SOFTWARE DEVELOPMENT PROJECTS IN NOKIA NETWORKS HELSINKI

The software development projects in Nokia Networks follow specific processes, common for all the projects carried.

6.1 Nokia Networks Processes

Three core processes of Nokia are:

- *Product Creation Process* is a value chain, which includes all the cross-functional activities of defining and developing new products, systems and platforms. These activities cover the development of the actual products as well as the development of the capabilities needed for marketing, producing, delivering and maintaining the products.

- *Delivery Process* is a cross-functional and modular process through which customers are provided with products, solutions and services. The objective of the Delivery Process is a satisfied, loyal and profitable customer who wants to do more business with Nokia.

- *Business Support (BS) process* describes the business specific process of creating, verifying, sharing and continuously managing strategic and shorter-term objectives and plans on the various organizational level of IMN.

In this study we will concentrate on Product Creation Process (Figure 8).

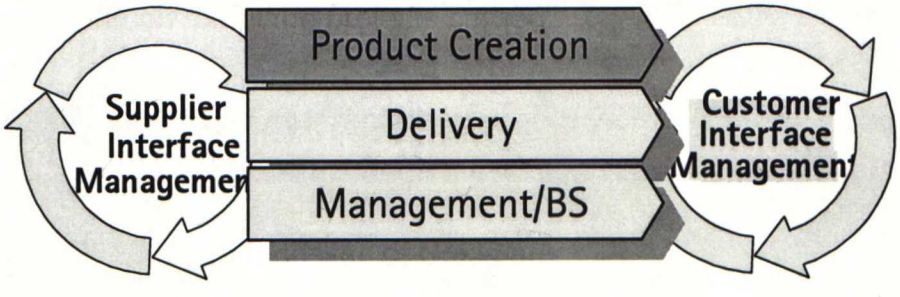

Figure 8: NET Product Creation process as a part of Core Processes (Märijärvi et al. 2002)

6.1.1 NET Product Creation Process

One product creation process model is designed and all product programs were to use that process template. But in real life product programs have different needs and priorities depending on the product and market situation. Competition on telecommunication markets is fierce and in order to succeed in that market environment it is essential to have processes that support and adapt to product program's needs. The NET Product Creation Process framework (Figure 8) was developed because the old model was unable to meet the different needs of different programs. The idea of the old model was to have common process model that would fit for all purposes. Now, however, different process variants are needed for different programs. NET PCP model gives more freedom for program managers and best process can be tailored for each program. NET PCP framework was created in large extent by the people that are going to use it. It also supports incremental development unlike the old model. (Märijärvi et al. 2002)

The NET PCP has process variants for different market modes. This means that different process model is used for product creation in different phases of market lifecycle. The traditional waterfall model (Main Street model) is still used in NET PCP framework, but it is aimed for late market phase. Incremental process model is used in early market phase, where it is essential to be able to react fast.

Incremental model can also be used in other market phases if capabilities of incremental process model are needed.

6.1.2 Product Creation Process Milestones and Phases

Product creation process is divided in phases separated by milestones. In order to move to next phase, milestone criteria must be fulfilled.

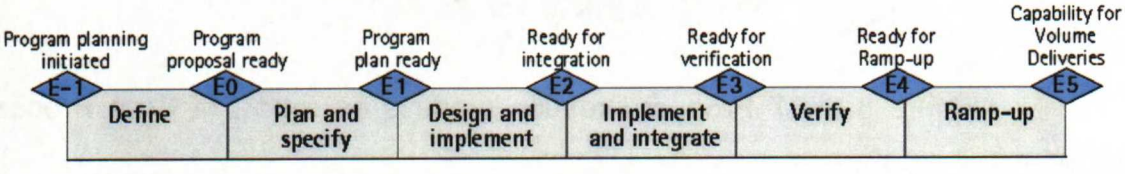

Figure 9. Milestones and process phases (Soininen et al. 2001)

Process phases are described briefly below:

Define:

After E-1 decision product program is allowed to use resources. In define phase product program is defined on a very general level. Product technical feasibility is assessed.

Plan and specify:

Product requirements are specified and feature candidates are defined. Product creation process model is selected.

Design and implement:

Product requirements are transferred into design. Product architecture is created and after that product implementation is started.

Implement and integrate:

Product implementation continues and integration is started. Integration means that product parts are put together and tested whether they work together as planned. Integration testing reveals problems with interaction and changes to implementation are done as needed to solve problems.

Verify:

Purpose of this phase is to verify that product implementation fulfils the requirements set in the planning phase. This phase includes functional testing, system testing and system verification activities. Trial deliveries are made to most important customers.

Ramp-up:

After product has been verified and validated, it can enter ramp-up phase, where volume deliveries are started.

Maintain product:

Product maintenance includes updates and fault corrections. Some modifications can be made to product in order to meet customers' changing needs.

Ramp-down:

When product becomes obsolete and is to be replaced by a new product, it enters ramp-down phase. No new deliveries are made and product support infrastructure is gradually run down. After ramp-down phase is completed, the product is retired.

6.2 Presentation Case Study

The program, which makes the object of this case study, is called "ETNA" and its scope object is release of the software product "ABC" version 1.0 (we refer to it as ABC 1.0).

The software program/project (even though the correct naming is "program", it is allowed to address it as a "project" considering that we will analyze mostly R&D and Testing related issues, which reduces the focus) proposed for study has as subject a new product to be developed in the Business Unit. This software product aimed to be a continuation of a 4 years old product. The rapid pace of market development and customers (in our case mobile operators) demands imposed that a new product is essential to keep the Business Unit and Nokia Networks in general on the top of operators' list. In the scope of the program was included:

- Enhance end-user satisfaction through focus on usability
- Change SW and HW platforms to be aligned with Nokia Networks core strategy;
- Address the customers' requirements by reducing the costs of the hardware equipment
- Reduce overall price of the product comparing with previous software products offered to customers and with the prices of the competitors

- Focus on increase in software quality

The program consists of the following projects:

- R&D Project: responsible for the design, implementation, unit testing and integration of the external software components.

- Customer documentation and training project: responsible for producing customer documentation and customer training material as well as taking care of technical competence transfer.

- Testing Project: responsible for overall test planning, functional testing, system testing and usability testing.

- Delivery Capability Creation project: responsible delivery capability creation, Sales Configurator tool creation (Product Manager), and SW production and distribution arrangements.

- Customer Service Capability Development Project: responsible for planning and organizing development of service capability competence in CS.

- Product Management Project: responsible for business cases, product specification, requirements management, product roadmap, input for logistics, and customer pilots.

- Marketing Project: responsible for planning marketing, product launch and sales promotion.

- Technical support project: responsible for verifying product installation and upgrading features and documentation.

The Program Manager heads the program. Each project specified above is leaded by a Project Manager, responsible for planning, resourcing and execution of the projects.

Overall Description

The program follows the iterative development process. The activities are shown in the following diagrams:

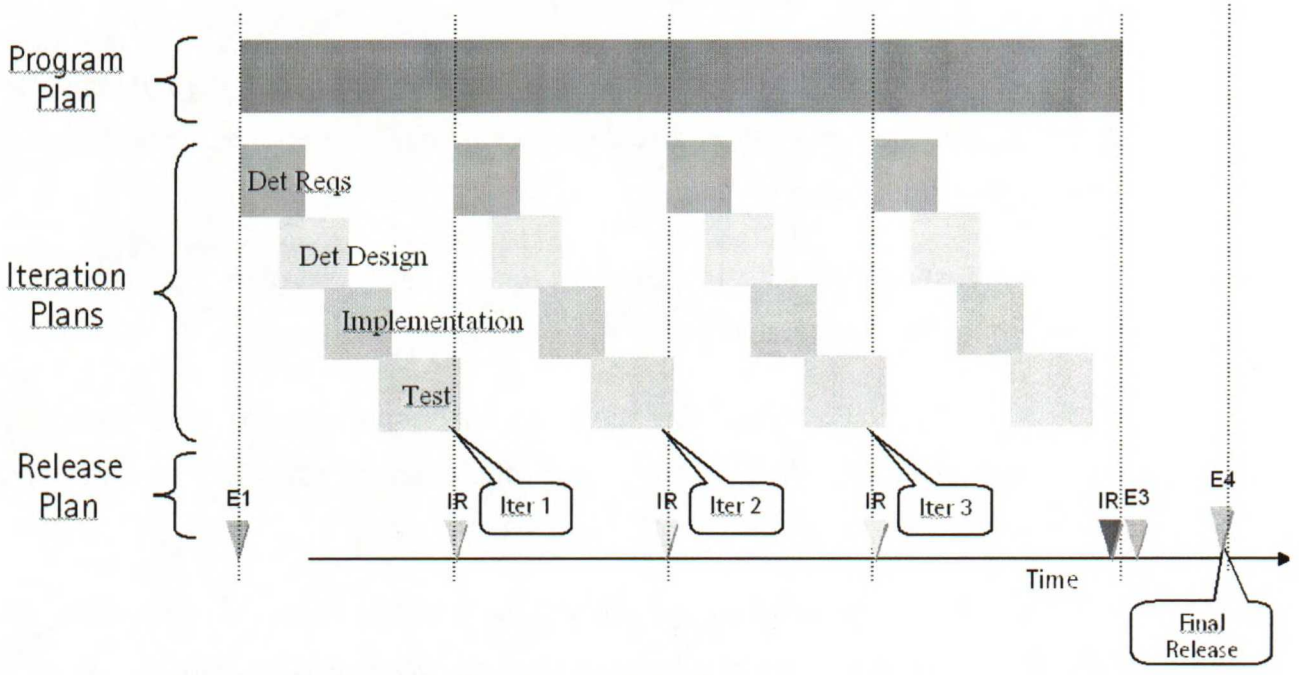

Figure 10. Iterative development model

The phase E1 – E3 is divided in multiple sub-phases (here iterations), each containing all the steps a development process involves: detailed requirements (high level requirements of the product are detailed so it makes easier for R&D to understand what is requested to be implemented); detailed design (the pre-implementation phase, in which R&D creates detailed design documents describing in detail how the implementation will be done), implementation phase and testing. Each iteration has definite quality exit criteria. The iterations are incremental, so that at E3 phase the whole software will be built up and tested by pieces. The real system testing phase commences at E3 and E4 milestone is granted only if the system meets the quality criteria set at E1 phase.

Staffing

Staffing will vary somewhat from iteration to iteration. However, one of the reasons for choosing iterative development model was the unavailability of the needed resources. There was no plan for acquiring new staff or getting new

subcontractors. The resources in the Business Unit were divided between the two current programs existing simultaneously in Business Unit. Due to this situation, it was considered that iterative model provided the chances to create the product step by step. Overall staffing plan is as follows

Nokia:

- R&D Helsinki: 12-15 persons
- R&D Pittsburgh: 10 persons
- 1 Test Manager
- Testing: 11 persons
- 1 Product Manager
- 1 Chief Architect
- 1 R&D project manager
- 1 Chief Engineer
- 1 Quality engineer
- 1 Competence transfer manager
- 1 Customer documentation project manager
- 1 Delivery Capability Creation Project manager
- 1 Marketing manager
- 1 Business development manager

Subcontractors:

- R&D: 3 persons
- Customer Documentation: 3 persons
- Testing: 3 persons

Effort Estimates

Effort estimates excluding subcontracting:

Discipline	**Planned person months**
Program and product management	38 man months
R&D	265 man months
Testing	87 man months
Customer documentation and competence transfer	18 man months
Net delivery and service capability, technical support	8 man months
Total	350 man months

Subcontracting effort estimates:

Discipline	**Planned person months**
R&D (X component)	6 man months
Testing	49 man months
Test automation	9 man months
Customer documentation	11.5 man months
Competence transfer	6.5 man months
Usability evaluation and testing	3 man months
Total	145 man months

Program Control and Tracking

The control and tracking of program tasks are done by various means.

1. PMT (Program Management Team) Meetings

PT will meet on a weekly basis. The projects will have their own meetings. These meetings are mentioned in the project plans. Program action points are registered in Action plan. The presence to PMT is mandatory for each project manager. The team leader for R&D Pittsburh participates via conference call.

2. Program Metrics

There are two sets of PCP metrics, such as follows: post E5 metrics (not in the scope of this study, but very important for further analysis) and program time metrics. The latter are:

- Effort Estimation Slippage (m)
- Number and Origin of the Content Changes (m)
- Faults Found before Customer Release (m)
- Test Execution Progress (m)
- SW Problem Report Correction Time (m)
- Time to Market (m) Degree of Reuse
- Release Slippage (m) Slippage of integration points
- Release Completeness (m) SW Build lead-time
- Faults Found before, during and after Integration
- Effort caused by content changes

3. Metrics follow-up

The above listed metrics are followed up monthly in the PMT; the Quality Manager does presentation. The metrics status is reviewed and possible corrective and preventive actions are taken. Action points are recorded into the programs action points log.

4. Status Reporting

All projects present a short status summary on their area in the weekly PMT meetings. These reports, together with the meeting minutes, will be distributed to all program personnel, and others on an as-needed basis.

The following best practices are used in the program:

- Efficient review of requirement specifications, design documents and test cases

Before R&D Requirements and Design Document is submitted for review, Quality engineer makes the first pre-check to ensure the readiness for review. For design documents 3-hour review sessions are set up and the 1st hour of the session is used by each individual to read through the document. After that, all comments are discussed and proposals made and documented. Same approach is used for testing cases preparation.

- Develop iteratively

Iterative development accelerates risk reduction. The first iteration should include the basic architecture that exposes the highest risk.

- Continuously verify quality

Quality targets set for each individual iteration and set as exit criteria. Faults are 100 to 1000 times more costly to be found after deployment than during testing planning phase. For this purpose, automation is used to a large extend.

- Manage requirements and change

Requirements and changes during the program are carefully evaluated and also their impact to product functionality and program timetable. Strict decision-making process is used and decisions are clearly documented.

- Inspection minutes

Milestone deliverables (including system specifications, customer documentation, etc) are reviewed and inspection minutes are prepared stating the participants, most important findings and their severity and action needed/taken.

- Modular design and much greater emphasis on unit testing
- Improved design templates and more detailed reviews
- Unit testing done during implementation phase, as opposed to afterwards -> unit testing was included in the schedule
- Use of automated test cases

5. Lessons learned

The program collects lessons learned data after each milestone and each iteration. The Q&P manager and the quality engineer arrange short sessions for gathering information on:

- Good practices, well done things
- Problematic areas, things to improve
- Process feedback

In major milestones (E1-E5), the main participants to the lessons learned sessions are the PMT members, in iteration milestones the focus is more on understanding the iterative development model from SW development & testing point of view, so the participants are defined iteration by iteration.

The findings in the lessons learned sessions are documented and communicated within the program and the business unit by the Q&P manager. Lessons learned findings are also included into the program final report by the

program manager. The process feedback is used as input for the continuous development of the processes used in the product line.

6.2.1 Risk Management Practices

Program Manager has the main responsibility for Risk and Critical Success Factor Management. Additionally, each project manager and product manager will closely follow and monitor the risks in their area of responsibility. Whenever the probability of the risks increases/decreases considerably, it is taken into the PMT meeting for discussion. For this purpose a risk log is available. The risk log is available in the Appendix 2. However, the main areas in which the risks are raised are Program level general risks (includes also Product Management), R&D and testing risks. As observed in the past programs, the other projects of the program (logistics, delivery, documentation) are not subject to high severity risks.

Risk Management activities commenced before E1 milestone of the project, when the high level planning and product level requirements are done and approved. Several risks were continuously raised by different projects and collected in a risk log discussed weekly in PMT meetings (presented in Appendix 2). The risks still open at E5 date passed to the future release or to Maintenance phase.

In the table below, we present some of the most important risks raised. We believe that these risks are common for the projects developed inside software companies. The continuously high demands of the market environment impose research for cost-reducing software platforms, improve time-to-market, improve development processes. The issue to be raised here is how good/experienced are the companies to tackle these risks and gain market share.

Risk ID	Area of occurrence in the program	Description	Milestone when the risk was raised
1	General	Changing of software development model from waterfall model to iterative model	E1
2	General (product management/marketing)	Fierce competition in this field of mobile business lead to more competitive products from competitors	E1
3	R&D and Testing	Business environment does not allow acquiring new resources and the resources available should be divided and/or redirected (often persons are involved in the two simultaneous programs)	E1
4	R&D and Testing	People are not trained in new SW and HW	E1
5	General	This product is conceptualized as a current complement for the one already on the market for 4 years and in the future a substitute for it. This situation implies many interferences between the programs. However, the roadmap includes another two releases of the old product until the new one is mature enough to satisfy all the market's demands.	E1
6	R&D and Testing	Delays due to implementation of new functionality requested by customers	Iteration 2
7	R&D and Testing	Departure of experienced team members	Iteration 2

8	R&D	One part of R&D project team is situated at another location (Nokia U.S.A)	Iteration 2
9	R&D and Testing	Totally new product and architecture	Iteration 3
10	General	The responsibility of one team of subcontractors (inside Nokia) passes to program	Iteration 3

Table 6. Risks description

The challenges once identified, the program management team recognized Risk Management as an area of most importance. This is the first program in the Business Unit to acknowledge the importance of an intensive risk management activity throughout the whole program and also to correlate these actions to the ones taken by parallel program(s). The expected benefits of risk management were:

- Effective strategic planning
- Better cost control
- Enhancing shareholder value by minimizing losses and maximizing opportunities
- Increased knowledge and understanding of exposure to risk
- A systematic, well-informed and thorough method of decision making
- Increased preparedness for outside review (upper management boards)
- Minimized disruptions
- Better utilization of resources
- Strengthening culture for continued improvement

- Creating a best practice and quality organization

6.2.2 Situation at the End of the Program

At the end of the program, the situation in the most significant areas was as follows:

- The product came out in the market with less functionality that expected at the beginning
- 40% of this functionality was different than the one initially planned.
- The iterations schedules were delayed with minimum 1 month per each iteration and the total delay of the program was 5 months.
- The total budget of the program outrun the initial budget with 30%
- The customers were overall happy with the product and 2 mobile operators (the ones, which agreed to pilot the product) are interested in buying it, even though the total cost was higher than it was advertised.
- The Business Unit decided to go on with second release. The decision was mostly based on the fact that the timing for this release was ahead to the situation on the market. According to the latest estimates, the market maturity for this type of product will be reached in 6 months or 1-year time.

The next part of this chapter describes the risk situation at the end of the program. Our analysis is based on the Table 6. We evaluate the status of the risks presented there, what were the measures taken to close them up successfully and also the motivation behind the ones still opened.

Risk 1: The status for this risk at the end of the program is still OPEN. Our analysis shows that the people did not adapt to the iterative model easy. Due to the changes in processes and procedure, many misunderstandings were created and wrong paths followed. People tended to use the same methods as for waterfall model, but they ignored the fact that iterative process requires adapting to more rapid pace of development and more planning and controlling activities.

The "lessons learnt" sessions and feedback questionnaires were not properly analyzed and used from the early phases of the project due to tight schedule and rapid passing from one iteration to another. The R&D processes took longer than expected, especially in the first two iterations. Because of this, the functionality implemented in the first iterations was too less comparing with the R&D effort for the last two iterations and much of the important functionality of the product was not developed from the beginning. This situation raised a number of risks later and also had a negative impact on testing schedule, which faced the fact that the effort in the first two iterations was much smaller than the estimations and also the resources were partially blocked and not been able to fully concentrate on testing the other product.

Risk 2: The status for this risk at the end of the program is CLOSED. The beginning of the program was marked by fear of threatening competition and a number of very demanding requirements were raised trying to overcome what the competition might be able to develop. However, this situation is discussed in the evaluation of impact of Risk 6. One significant note to be written here is the poor description of the risk.

Risk 3: The status for this risk at the end of the program is still OPEN. It became very hard to divide the resources between two on-going programs in the phase of full development. The people became confused and could not set right the priorities. The blocking of testing resources in the first two iterations was caused by an error in feature planning and finally had impact not only on delays in testing schedule at the end of the last iteration, but also affected the planning and execution of testing for the other product, which was in a full testing phase. More resources would have meant finish testing earlier and deliver the product to the customers and also start the full work on the next release (the one planned to be compatible with ABC 1.0)

Risk 4: The status of this risk is CLOSED. Training took place and the level of competence required was satisfactory.

Risk 5: The status of this risk at the end of the program is still OPEN. The coordination between the two programs was not as successful as hoped. One area of the problems raised was presented in analysis of Risk 3. The delays in both

programs made difficult to correlate the release dates and therefore one program was obliged to drop some requirements in order to meet the same deadlines. No person was allocated to coordinate the resources and product requirements for the two programs.

Risk 6: The status of this risk is CLOSED. Even though the decision has been made to close the risk and change the status to *resolved*, we think that a thorough analysis of it is required. It is very common for software companies to forerun the market demands. This occurs essentially because of the business environment where these companies act, which is very challenging and in continuous movement. If one company is not one step before the market and the competition, the risk of loosing market share is very high. This is one reason for which the software programs should be very flexible in adopting requirements. In the software development processes, the generation of requirements usually finishes at E1 milestone, but the current circumstances makes this impossible, particularly in the fairly medium to large projects. However, for a good progress of the projects, the requirements (especially after E1 and even after the first iterations) should be more carefully evaluated, as it is likely that their impact is deeper at this stage. The situation in the program was so that 20 requests were made per month for new functionalities or changing in the already existing requirements. The process of analyzing these change requests took time and effort from both program management team, but also from R&D and testing team members.

Therefore, the program we analyze here demonstrated a poor management of requirements, which were changing all the time and created confusion in R&D and testing teams. Even though at the end, the customers were reasonably content with the product features and delivery schedule, the approach taken was not good and we consider a pure matter of luck that the program did not end in a completely unsuccessful situation.

Risk 7: The status of this risk at the end of the program is still OPEN. This risk is very common to all the companies, software oriented or not. The movement of job market is well-recognized element of the business environment at the present time and in the large companies, due to the inevitable bureaucracy, these issues are difficult to tackle. Building the same level of competence as before for

new members takes time and training. It was not possible to achieve this during one program length, but nonetheless the quality of the software was satisfactory.

Risk 8: The status of this risk at the end of the program is still OPEN. The coordination of the R&D project was not successful. The communication between members of the same team could be burdened by the differences in location, especially because of time differences, different cultures and ways of working. Although the fact that the organization is the same, the processes and procedures are the same is an advantage for virtual projects, many problems were raised by confusion in requirements and design phase. The administration and database system used were unique as well as the development tools. The good conditions created are a plus for the project management team and despite all these, we believe that the problems created were mainly due to non-communication and non-understanding in the same manner the requirements and ultimately the functionality to be implemented. The defects raised in testing phase also were very much disputed and their resolution was delayed because of this, causing at the end disagreements inside the team. We will accentuate this topic in next chapters.

Risk 9: The status of this risk at the end of the program is still OPEN. The risk was poorly defined in the sense that it contained several general areas to address and it should have been distributed in several risks. One reason for changing the architecture of the new product was to improve performance at the same time as the cost of the equipment decreases. Unfortunately, this did not happen to some extent because the qualities of HW equipment were not completely investigated if they correspond with the SW necessities. The next programs should attempt to solve the trouble. Also, the new architecture caused problems for R&D in designing the software system and hence delays comparing with plans. The poor definition of system requirements also had a role for raising the importance of this risk.

Risk 10: The status of this risk at the end of the program is still OPEN. The responsibility for the core component of the product passed in the middle of the program to R&D team. A new competence should be built, fact that required first of all time. The passing was done smoothly and the former subcontracting partner delivered to the Business Unit the part of software agreed but the time for

acceptance testing was less comparing with the functionality involved and by the time the whole system was integrated, the responsibility for any defects found passed to R&D team. At the end, the required competence could not be entirely built and so the product was delivered to the customers with less functionality than expected.

In this chapter, we have studied the unrolling of a software development project inside one of the Business Units in Nokia Networks. We have described first the general processes and models for software development used throughout Nokia and then, go into further details thru the particular project presented. The focus in this presentation was on risk management procedures adopted by the management team and how were they confronted up to the end of the program.

Primarily, our intention was to reveal that the program management team followed the reputable processes inside the company, processes that proved to be successful and are in a continuous improvement in order to achieve all the time the best practices in software projects. At the general level, the program was well coordinated by the management team, all the projects were intensely involved in the good unfolding of the entire program and the quality and management processes were applied successfully. The controlling and reporting phases proceeded as planned and the whole team was involved in constant improvement of practices of work.

We consider nevertheless that some activities still let place for enhancements. The activities in question are the survey/feedback practices engaged in the lessons learnt sessions, management of requirements and customers demands and risk management measures.

Even if the approach taken for iterative model was not entirely successful, we may conclude that this project confirmed the advantages of using this software model in the changing and rapid environment of software companies. This study has been proven that we are living into process-centered SW environment, where organizations are focusing on improving their SW development processes. This chapter summarized the main drivers and characteristics of a successful SW development process, in which the new project is producing innovative, interesting and leading product for the customer with the agreed price and short lead-time. Firstly, the process should support business goals and objectives and the

organizations roles and responsibilities should be defined to enable and support the process. Management practices should be defined and enforced to monitor and support the process. Skills should be acquired to enable effective performance of the process tasks. And finally, tools and technology infrastructure should be designed and built in such a way as to automate and enable the efficient performance of the process tasks.

Due to the goal of our study, we will center upon the risk management procedures with references to two major issues of present interest in project management:

- Management of requirements. As we will demonstrate in the next chapter, the activity of establish the requirements as clear as possible has a great beneficial impact on the well unfolding of the software program.

- Management of virtual projects. Virtual projects solution to software development projects is more and more a desirable approach to project work for the reasons described in the previous chapters. In the same time, managing of a virtual project is a very challenging task and make it successful depends on some factors of influence: team leader capabilities, qualities of shared equipment, communication issues, etc.

6.2.3 Analysis of Risk Management Activities

The risk activities will be analyzed starting from the key risk management activities identified in Chapter 4.2 and based mostly on SEI's risk taxonomy, presented in Chapter 2. In creating the framework for analysis, elements from the other models are to be used, but SEI's risk model was considered to be the most appropriate and comprehensive.

In this chapter, we try to analyze the situation of the project presented in the previous chapter through the perspective of the key activities identified as fundamentals of risk management and furthermore to improve their viewpoint based on the assessment of the studied project.

Risk Management Start Up phase commenced by identifying the need for exhaustive attention on risk management, pointing out the most probable areas with high level risks and the owners for each. As mentioned in the previous chapter, Program Manager was appointed the coordinator of risk activities. Project managers were responsible for raising/monitoring/solving the risks in their projects as well as announcing the PMT weekly about the status with new/old risks. The methodical analysis of risks was planned for each iteration and at E-milestones. The planning phase of risks evaluation has the meaning of assigning responsibilities and establish the reporting policy.

Goal and Stakeholder review phase usually takes place before E1 milestones also and it is reviewed throughout the whole program. In this case study and many of the projects, this analysis is part of Business Case prepared by the product management team. Even though it is part of the development of one program, we believe that it should also be a constitutive part of risk management procedure.

We can take as example our case and propose the key stakeholders identified and prioritized as follows:

Priority 1: Nokia customers, which already have in use the previous software product produced by this Business Unit. They are the "must win" customers and the efforts should be directed first of all to satisfy their demands.

Priority 2: New customers, who have to be convinced to buy the two products together.

Priority 3: Other Nokia Business Units, which produce software for mobile operators and whose products, ABC 1.0 must be compatible.

Other stakeholders were identified as: R&D and testing teams, marketing and sales departments, logistics.

The goals for each stakeholder are carefully analyzed. We will present them shortly. The goals identified for the existing customers and new customers are: achieve enhanced technology, increase end users satisfaction and reduce cost/transaction for the entire solution provided. The goal recognized for the other

Nokia Business Units was: develop compatibility between the products in the intended time scale. For the internal stakeholders, amongst R&D team goals are: speed up the documentation process (develop better processes and procedures for creation of detailed requirements and design for product features), increase product quality, delivery the builds in time for testing. Testing team goals were given special attention: increase product quality through better methods and testing tools; finish testing in time according to plans.

The emphasis on the goals is significant because it gives valuable input in assessing and prioritizing the risks and help in decision making process in the later phases and we believe that the lack of reviewing the stakeholders goals in the risks evaluation can pose crucial problems in the good development of the entire project.

Risk identification phase is a logical continuation of the previous phases. Several risks were acknowledged during the risk assessment meeting (program manager and all the project managers) and a first high level risk list (named risk log) is created. This activity can be improved by determining in the same time with risk identification, the areas and the cycle phase when the risks are likely to appear.

Among all the risks phases, we consider as the most important risk analysis phase. Our research showed that in most of the cases, risk evaluators fail to a give this activity the right significance. The same thing happened with the case we study. Indeed, the evaluators made the risk prioritization and analysis, established the owners and impact, but they did not succeed in giving the appropriate priority to the review of risks root cause. We believe that identifying first the cause of occurrence of one event (may it be risk or anything else) can save a lot of effort in controlling it. They have also failed in creating a risk scenario for the most relevant risks. Risks scenarios contribute to thorough analysis and identification of the best paths for controlling and monitoring the risks, not to mention that they can save a lot of time and efforts from the team members.

Risk Control activities depend largely on the methods and means each company has to prevent the risks from materializing. During this phase, analysis of risk impact plays an important role. This analysis is based on the risk prioritization done in analysis phase, which at its turn should take into account very much the importance of stakeholder goals review.

To take again as example our study, we believe that program management team did not pay enough attention to these two phases. Risk log (Appendix 2) recorded demonstrated an interest and awareness of the situation, but the fact that no deeper analysis took place made impossible closure of the majority of the important risks. Superficially, all the risks were addressed but not analyzing profounder the causes and the true impact as early as possible proved to be an omission that a project usually cannot afford. Many of the problems could have been tackled and at least partially solved if more consideration would have been paid to communication issues inside the teams located at different sites or better definition of requirements.

Risk monitoring activity was in place and it was constantly followed, by analyzing risk situation twice per month in PMT meetings and assigning the right responsibilities to the owners. Maybe one plus of attention to present the risk situation to the entire project team would have helped in sorting the issues out more quickly.

To summarize the situation with the project in case, there are at the end of the program 110 risks, as follows:

- 22 risks raised at Program General level
- 58 risks raised from R&D project
- 30 risks raised from Testing project

Of this total, 15 risks were still in state OPEN at E5 milestone (Mass Delivery of the product). In this document, we only presented the general and high-level risks of the program, which we considered most relevant to the object of the thesis. It is also important to mention that among R&D risks raised, a total number of 25 were related to the collaboration and deliveries between the two teams located in Finland and U.S.A. Equally important to our observations from this work it is the fact that from 22 Program level risks, 10 were associated with definition and continuous changing of requirements (both customers and system related requirements).

7. STUDY PROPOSAL

This chapter contains the schema we propose for a comprehensive analysis of risks in the context of software development projects.

The model here is mostly based on the key risk activities identified in Chapter 4.2 and the outcome of our empirical study. The results of combining the findings extracted from the theory available with the study we have pursued are here a cyclic model for initialization and record the risk management activities. In the framework presented below we also included the key decisions involved in development of a project like choosing the software development model as well as going for virtual teams or not. We believe that these decisions should be highly influenced by the risks identified and their impact on the stakeholders' goal.

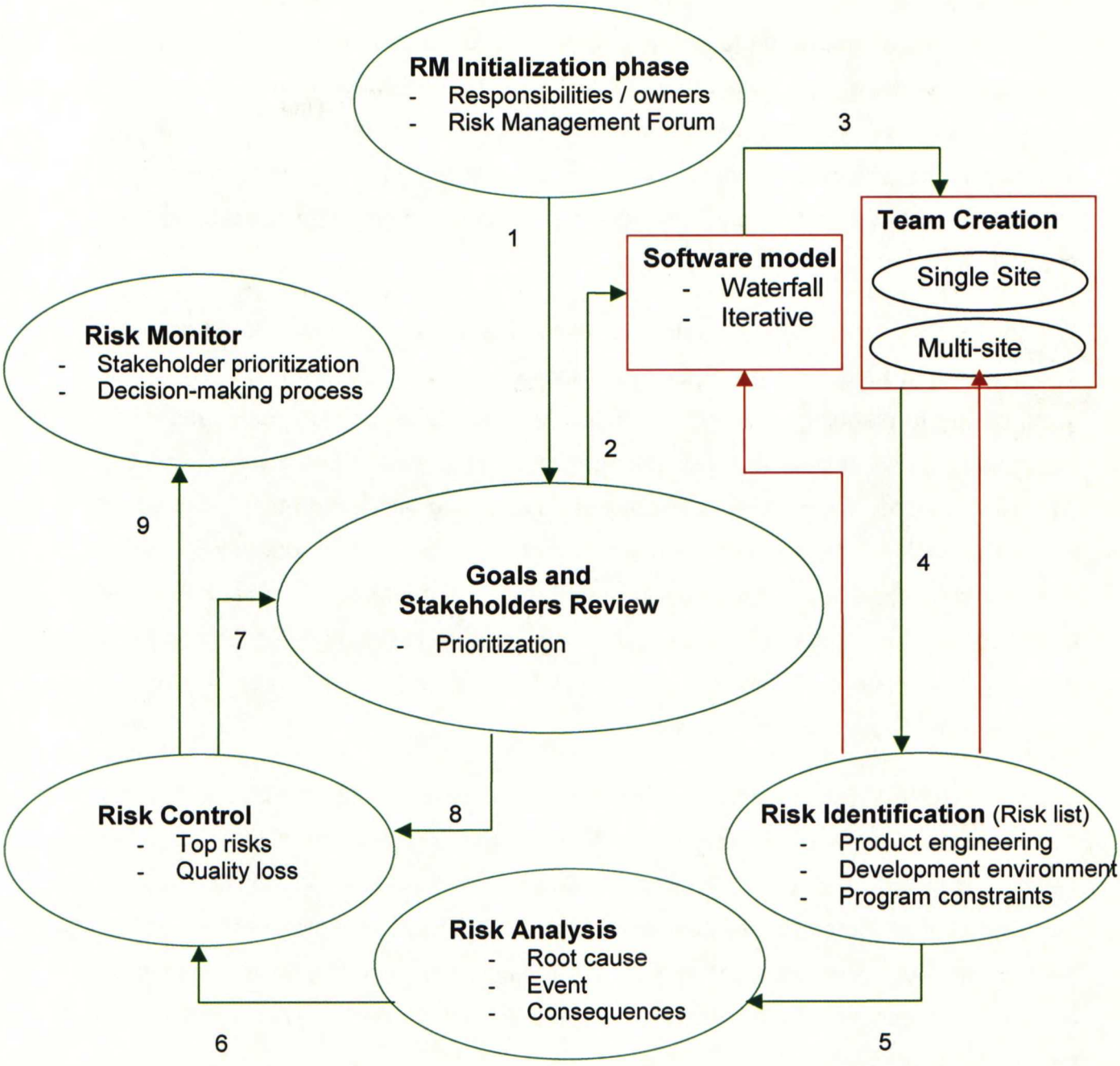

Figure 11. Risk Management workflow

Risk Management Initialization (start-up) phase should start before E1 phase, when all the decisions and high level planning is prepared. Its first and essential activities should be to assign decide the procedures about the ownership of risks and assigning responsibilities. Also, Risk Management Forum is created. Usually, it includes program management team and the owner of risk management activities is the program manager. For large projects with high-risk implications and strategic importance, the responsible person may be a dedicated person named Risk Manager.

Once basic activities are identified, Stakeholder Review should be held. The analysis is based on the Business Case (for external stakeholders) and high-level program planning. It contains evaluation of stakeholders' goals and their prioritization. This review is most valuable input for prioritization of risks and the corrective actions. The study of stakeholders and their goals should be perceived as one of the basis for the program management team to choose the adequate software development model (e.g. if the requirements from the customers cannot be clearly identified and the needed resources are not available, iterative model is the choice; if the program schedule is not very tight and the high priority is a stable product, waterfall model could be the right option).

Bearing in mind these phases, the management team can analyze the options (if it is the case) of going for multi-site distributed projects or not. We have decided to include the reference to multi-site projects in this model considering the fact that in the contemporary software business, software development programs have most likely one virtual project or sub-project. We believe that the impact of distributed teams should be deeply analyzed in the context of risk and project management activities.

Therefore, the decision of going for virtual teams or not should be made on the level of competence required, resource allocations, uniqueness of the software tools, availability of communication means and naturally overall savings costs. The benefits and risks expected should be put in balance and coordinated with the overall situation of the program.

Risk Identification is the next step and consists of creating a risk list. The risks are identified and described. Risk List expands the understanding of risk

environment crucial for the success of the project. Risk List as such is only a list, which does not include or require any further analysis at this point. The areas generating risks are acknowledged and included in the list (SEI's Risk Taxonomy makes the most comprehensive separation of activities).

Risk Identification phase is the basis for Risk Analysis, which is a more systematic investigation of the risks raised. Many of the programs' decisions can now be supported by the risks analysis, by creating realistic risk scenarios, which contain: risk root cause, risk event (description of risk) and predicted consequences.

Risk Controlling activities may need to be continuous and they can be implemented in and between related processes. Controlling activities can also be planned as separate actions that do not have a permanent place in the process but can be made only for a specific situation. During this phase, the risks are prioritized for resolution based the expected effect (evaluated in Analysis phase) on the achievement of the stakeholders' goals. This phase also has the role of quantifying the losses to be generated occurrence and impact of the risks.

Risk Monitoring phase involves principally the activities of communication, escalation and partially decision-making method regarding risks identified and chosen for controlling. Management team typically monitors top risks chosen for controlling. For the risks with very high impact, the input from stakeholders is necessary to track the corrective actions.

Alongside with this schema, a risk log is generated to be able to keep track and follow easier the risk situation. A comprehensive and up to date risk log helps in developing the capabilities of synthesizing and solving the situations in a desired manner. In Appendix 3, there is the model of risk log we propose. It is based on the information presented throughout this study.

The findings of the present chapter and also the entire work are designed to tackle the risk management actions first of all, with the scope of easier integration of risk management in the current activities, which take place within a software project. The external and internal environment in which a company acts, with everything that they involve mandates the need for such integration. This means jobs market, competitors, technology, social and economical issues to name just a

few. Every element of this circle can be a risk generator area and success or failure depends enormously on the strategy adopted by each company or project.

8. CONCLUSIONS

In our study, we have followed a software project from different perspectives: the process cycle, development model and distributed environment perspectives, trying to identify the areas that are most likely risk enablers or on the contrary the areas recognized as low degree risk generators.

Software lifecycle encompasses the time from the decision to start developing software until the moment when the software pushed out of service. Amongst the lifecycle phases we mostly identified *system requirement study, software requirements study and requirement analysis* as the first area that rises problems and risks.

System requirement study

Software is always part of a larger system. Therefore work begins by gathering requirements for all system elements. This system-wide view is necessary when software must interface with other elements like hardware and databases. In the case of embedded systems, this phase is often called system engineering. The system is thought as a whole while the architecture of the system as well as responsibilities and functions of different system components are defined. The purpose of this phase is to set general, system level requirements, which are often called *customer requirements* because they define customer's needs, but do not define what kind of system is going to fill those needs. The biggest challenge of this phase is to find out and thoroughly understand what customer's needs really are. If customer's needs are improperly interpreted, it is impossible to build a good system.

Software requirements study

Collected system requirements are analyzed and software requirements are derived from them by figuring out what kind of software is going to meet the system requirements.

Requirement analysis

After the overall software requirements are known, they are analyzed to produce the detailed requirements. Requirement analysis and specification is all about translating customer requirements into exact software requirements.

Designing and implementation phase is all about interpreting the requirements correctly. The communication between teams is very important at this stage, especially when distributed teams are involved. Design exerts a major influence on other process phases. Design has an iterative nature: a solution is created, modeled, evaluated and after a few of these cycles, turned into specification for implementation phase.

Testing is usually done on multiple levels (module testing, integration testing, functional testing, system testing, acceptance testing). Especially in research and development projects testing, error tracking and error correction can form a significant part of the total costs of the system.

Maintenance

Work on software does not end after it has been released. Software often needs to be changed after it has been delivered to customer. This phase of software lifecycle is called maintenance. Maintenance includes error correction, adding some new features to existing software or possibly adapting existing software to accommodate changes in its external environment. Software maintenance takes care of these tasks by re-applying each of the preceding phases to the existing product.

Choosing the software development model is an essential decision. Most of projects chose lately the iterative development model instead of waterfall model due to its flexibility in adapting the software faster to the market new demands even during the same program. This flexibility encourages risk reduction and better management of risks.

The introduction of incremental process model has caused many positive effects in software development project in general, as well as in testing phase of the

project. The realized benefits include early detection of errors, reduced risks, faster feedback from the process, closer interaction between project teams and the possibility to change requirements during the project. On the other hand, also negative effects were detected, such as increased workload, more complex project management tasks and more exhausting work.

The third perspective we touched is the deployment of distributed environment and the advantages and disadvantages it brings on. The basics of multi-site development were fundamentally presented in Chapter 3.4. Software literature is acquainted with the success stories of virtual projects, but it abounds in failure examples. The case we studied recognized the benefits of this model, but it also confronted high risks due to: communication gasps, extra-work, things fallen in between, mistrust and tension between people in the different sites. Experience proves that it should be taken carefully into account to train people to work in multi-site, multi-cultural environment, to provide team building to cooperating teams in different sites and meet face-to-face as often as possible, especially during the requirements and design phase.

Combining successfully these areas is the fundamental assignment of project/program management team of any software project. But the achievement of such target is essentially dependent of the risk management procedures adopted.

Risk management and project management literature is rich in frameworks and analysis, theoretical and empirical examples. Therefore, we didn't pursue creating a totally new one since we neither have enough competence or experience to be able to do so. We have just tried to improve the existing ones by combining what we consider their powerful attributes. And on the top of this framework, we added the empirical observations from the real case study and our own thinking. We believe that our approach is first of all simple but credible and the interpretations are close to the reality of many software companies acting in the contemporary business environment. We proposed a practical framework that can be easily adapted to the requirements of different software projects no matter the size.

We acknowledge that the area of risk management is large and the analysis of different methods and methodologies to diminish or eliminate to some extent the risks from our business environment could be so much expanded. For

further studies of risk management in software development projects, we would like to propose using extensively the Utility loss theory. We have encountered the need for it in various aspects during the work for this thesis, but we decided to leave it out, as it would have changed the focus of this study entirely.

9. REFERENCES

Armstrong, S. 2001. Engineering and product development management: The holistic approach. 1st ed. Cambridge, Cambridge University Press.

Baskerville, R. and A. T. Wood-Harper. (1996) "A Critical Perspective on Action Research as a Method for Information Systems Research," *Journal of Information Technology*, (11) 3, pp. 235-246.

Berkeley, D. & de Hook, R. & Humphreys, P. 1990. Software Development Project Management – Process and Support. West Sussex, England, Ellis Horwood Limited.

Berny, J. & Townsend, P. R. F. 1993, "Macrosimulation of project risks -- a practical way forward", International Journal of Project Management, vol. 11, no. 4, pp. 201-208.

Bezirkan, A. & Mulazzani, M. "Experiences with Risk Management in a Large Multi-Site Project", in Proceedings of the Third SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Briand, L. C., Basili, V. R., & Hetmanski, C. J. 1993a, "Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components", IEEE Transactions on Software Engineering , vol. 19, no. 11, pp. 1028-1044.

Briand, L. C., Thomas, W. M., & Hetmanski, C. J. 1993b, "Modeling and Managing Risk Early in Software Development," in Proceedings of the 15th International Conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, pp. 55-65.

Boehm, B. W. 1981, Software Engineering Economics, Prentice Hall, Englewood Cliffs, N.J.

Boehm, B. W. 1987, "Industrial Software Metrics Top 10 List", IEEE Software, vol. 4, no. September, pp. 84-85.

Boehm, B. W. 1988, "A Spiral Model of Software Development and Enhancement", IEEE Computer, vol. 21, no. 5, pp. 61-72.

Boehm, B. W. 1989, Tutorial: Software Risk Management, IEEE Computer Society Press.

Boehm, B. W. & Ross R 1989, "Theory W Software Project Management: Principles and Examples", IEEE Transactions on Software Engineering no. July, pp. 902-916.

Boehm, B. W. 1991, "Software Risk Management: Principles and Practices", IEEE Software, vol. 8, no. 1, pp. 32-41.

Boehm, B. W. 1992, "Risk Control", American Programmer, vol. 5, no. September, pp. 36-43.

Bowers, J. A. 1994, "Data for project risk analyses", International Journal of Project Management, vol. 12, no. 1, pp. 9-16.

Carr, M. J., Konda, S. L., Monarch, I. A., Ulrich, F. C., & Walker, C. F. 1993, Taxonomy-Based Risk Identification, SEI Technical Report SEI-93-TR-006, Software Engineering Institute, Pittsburgh, PA.

Caplan, M. A. "Risk Management in Practice", in Proceedings of the Third SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Charette, R. N. 1989, Software Engineering Risk Analysis and Management, McGraw-Hill, NewYork.

Chittister, C., Kirkpatrick, R. J., & Van Scoy, R. L. 1992, "Risk Management in Practice", American Programmer, vol. 5, no. September, pp. 30-35.

Chittister, C. & Haimes, Y. Y. 1993, "Risk Associated with Software Development: A Holistic Framework for Assessment and Management", IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 3, pp. 710-723.

Conrow, E. H. 2000, Effective Risk Management: Some Keys to Success, Amer Inst of Aeronautics.

Conrow, E. H. & Shishido, P. S. 1997, "Implementing Risk Management on Software Intensive Projects", IEEE Software, vol. 14, no. 3, pp. 83-89.

Deutsch, M. S. 1991, "An Exploratory Analysis Relating the Software Project Management Process to Project Success", IEEE Transactions on Engineering Management, vol. 38, no. 4, pp. 365-375.

DoD 1988, Military Standard, Defense System Software Development, DoD-STD-2167A, Department of Defense, U.S.A., Washington, D.C.

Dorling, A. 1993, "SPICE: Software Process Improvement and Capability dEtermination", Information and Software Technology, vol. 35, no. 6/7, pp. 404-406.

Dorofee, A. J., Walker, J. A., Alberts, C. J., Higuera, R. P., Murray, T. J., & Williams, R. C. 1996, Continuous Risk Management Guidebook, Software Engineering Institute, Pittsburgh, PA.

ESA 1991, ESA Software Engineering Standards, ESA PSS-05-0 Issue 2, 2 edn, European Space Agency, Paris.

Eslinger, S., Ellis, C. M., Hoting, S. K., & Walden, G. F. "PACE System Risk Analysis: An Application", in Proceedings of the Second SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Fairley, R. 1994, "Risk Management for Software Projects", IEEE Software, vol. 11, no. May, pp. 57-67.

Fisher & Fisher "The Distance Manager – A Hands-On Guide to Managing Off-Site Employees and Virtual Teams", 2001.

Foo, S.-W. & Muruganantham, A. "Software risk assessment model", in Proceedings of the 2000 IEEE International Conference on Management of Innovation and Technology IEEE, pp. 536-544.

French, S. 1986, Decision Theory: An Introduction to the Mathematics of Rationality, Ellis Horwood, Chichester.

French, S. 1989, Readings in Decision Analysis, Chapman and Hall, London.

Friedman, G. J. "Risk Management", in Proceedings of the Second SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Gemmer, A. & Koch, P. "Rockwell Case Studies in Risk Management", in Proceedings of the Third SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Gemmer, A. 1997, "Risk Management: Moving Beyond Process", IEEE Computer no. May, pp. 33-43.

Greer, D., Bustard, D. W., & Sunazuka, T. "Prioritization of System Changes using Cost-Benefit and Risk Assessment", in Prroceedings of the Requirements Engineering Conference pp. 180-187.

Groth, J. C. 1992, "Common-sense Risk Assessment", Management Decision, vol. 30, no. 5, pp. 10-16.

Haikala, Ilkka & Märijärvi, Jukka 1998. Ohjelmistotuotanto. 6th ed. Jyväskylä, Suomen ATK Kustannus Oy.

Hall, E. M. 1995, Proactive Risk Management Methods for Software Engineering Excellence, Florida Institute of Technology.

Hall, E. M. 1998, Managing Risk: Methods for Software Systems Development, Addison-Wesley Pub Co., Reading.

Hefner, R. "Experience with Applying SEI's Risk Taxonomy", in Proceedings of the Third SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

IEEE 1987, IEEE Standard for Software Project Management Plans, Std 1058.1-1987, IEEE, New York.

IEEE 1992, IEEE Standard for a Software Quality Metrics Methodology, IEEE Std 1061 - 1992, IEEE, New York.

ISO 1991a, Information Technology Software Life Cycle Process ISO/IEC(JTC1)-SC7.

ISO 1991b, ISO 9000-3, Guidelines for the application of ISO 9001 to the development, supply and maintenance of software, ISO 9000-3:1991(E), International Standards Organization.

ISO. SPICE: Baseline Practices Guide, an unfinished draft of a standard being developed for ISO, version 1.00. 1994.

ISO 1998b, Information technology. Software process assessment. A reference model for processes and process capability, ISO/IEC TR 15504-2:1998 edn, International Standards Organization.

ISO 1998c, Information technology. Software process assessment. Concepts and introductory guide, ISO/IEC, ISO/IEC 15504-1:1998.

Jacobson, I. & Booch, G. & Rumbaugh, J. 1998. The Unfied Software Development Process. Reading, Addison-Wesley Longman Inc.

Jakobsen, Allan. 1998. Bottom-up Process Improvement Tricks. IEEE Software, January-February 1998.

Jones, C. 1994, Assessment and Control of Software Risks, Yourdon Press, Englewood Cliffs.

Kahneman, D. & Tversky, A. 1973, "On the psychology of prediction", Psych.Rev. no. 80, pp. 237-251.

Kahneman, D., Slovic, P., & Tversky, A. 1982, Judgment Under Uncertainty: Heuristics and Biases, Cambridge University Press, New York.

Karolak, D. W. 1996, Software Engineering Risk Management, IEEE, Washington, DC.

Koch, G. R. 1993, "Process Assessment: The 'BOOTSTRAP' Approach", Information and Software Technology, vol. 35, no. 6/7, pp. 387-403.

Kontio, J. 2001, “Software Engineering Risk Management - A Method, Improvement Framework, and Empirical Evaluation”, published by Laatukeskus.

Kerzner, H. 2003. Project Management – A Systems Approach to Planning, Scheduling, and Controlling. 8th ed. Hoboken, New Jersey, John Wiley & Sons, Inc.

Kerzner, H. 2000. Applied Project Management – Best Practices on Implementation. New York, USA, John Wiley & Sons.

Laitinen, L., Kalliomäki, S., & Känsälä, K. 1993, Ohjelmistoprojektien Riskitekijät, Tutkimusselostus N:o L-4, VTT, Tietojenkäsittelytekniikan Laboratorio, Helsinki.

Langer, E. J. 1975, "The Illusion of Control", Journal of Personality and Social Psychology, vol. 32, pp. 311-328.

Lipnack and Stamps, "Virtual Teams", 1997.

Madachy, R. J. 1997, "Heuristic Risk Assessment Using Cost Factors", IEEE Software, vol. 14, no. 3, pp. 51-59.

Madhavji, N. H., Höltje, D., Hong, W., & Bruckhaus, T. F. "Elicit: A Method for Eliciting Process Models", in Proceedings of the Third International Conference on the Software Process IEEE Computer Society Press, Washington, 1997.

Martin, Robert C. 1999b. Iterative and Incremental Development. C++ Report, April 1999. http://www.objectmentor.com/resources/articles/IIDII.pdf.

McCaugherty, D. "Criticality Analysis and Risk Assessment (CARA)", in Proceedings of the Third Annual Conference on Software Acquisition Management Technology Training Corporation, Washington, DC, pp. 306-340.

McConnell, Steve. 1998. Software Project Survival Guide. Redmond, Microsoft Press.

Meyers, D. J. & Trbovich, D. R. "One Project's Approach to Software Risk Management", in Proceedings of the Second SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Michaels, J. V. 1996, Technical Risk Management, Prentice Hall, Upper Saddle River, NJ.Miller, W. B. 1997.

Miller, W. B. 1997. Fundamentals of Project Management. In: Thayer R. H. (ed.). Software Engineering Project Management, Second Edition. Los Alamitos, California, USA, the Institute of Electrical and Electronics Engineers, Inc.

Monarch, I. A., Konda, S. L., & Carr, M. J. "Software Engineering Risk Repository", in Proceedings of the 1996 SEPG Conference Software Engineering Institute, PIttsburgh, PA.

Morin, J.-M. "Risk Driven Project Management: A Practical Approach", in Proceedings of the Second SEI Conference on Software Risk Management SEI, Pittsburgh.

Newland, K., Mays, M., Chapman, C., Gerdes, R., Hillson, D., Rawlings, P., Norris, C., & Vose, D. 1997, Project Risk Analysis and Management Guide, The Association for Project Management, Norwich Norfolk, U.K.

Pandelios, G. "Software Risk Evaluation and Team Risk Management", in Tutorial Presentations at the 1996 SEPG Conference Software Engineering Institute, Pittsburgh, PA.

Pandelios, G., Rumsey, T. P., & Dorofee, A. J. 1996, "Using Risk Management for Software Process Improvement", in Proceedings of the 1996 SEPG Conference SEI, Pittsburgh.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. 1993a, Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, SEI-93-TR-024.

Pentti Marttiin, Jari A. Lehto, Göte Nyman “Understanding and Evaluating Collaborative Work in Multi-Site Software Projects – A Framework Proposal and Preliminary Results”, 2002.

Petroski, H. 1985, To Engineer is Human: The Role of Failure in Successful Design, Macmillan.

Pfleeger, S. L. 1998, Software Engineering - Theory and Practice. New Jersey, USA, Prentice-Hall. Inc.

Phillips, D. 2000. The Software Project Manager's Handbook – Principles Thet Work at Work. Los Alamitos, California, The Institute of Electrical and Electronics Engineers.

Powell, Anthony. 1998, Strategies for lifecycle concurrency and iteration – A system dynamic approach. The Journal of Systems and Software, Issue 46/1999.

Pressman, R. S. 2001. Software Engineering – A Practitioner's Approach, 4th Edition, New York, USA, The McGraw-Hill Companies.

Pressman, R. S. & Ince, D. 2000. Software Engineering – A Practitioner's Approach, European adaption. 5th ed. McGraw-Hill. 915 p.

Rahikainen, M. 1998. Sulautettujen järjestelmien ohjelmisto-tuotantoprosessin tehostaminen yrityksessä. Oulu, University of Oulu, Department of Electrical and Information Engineering. Licentiate Thesis. Pehr Brahen julkaisusarja, julkaisunumero 1.

Reel, J. S. 1998. Critical success factors in software projects. IEEE Software, Vol. 3, No. 3.

Ricci, P. F., Sagan, L. A., & Whipple, C. G. 1981, Technological Risk Assessment, Martinus Nijhoff Publishers.

Rook, P. & Cowderoy, A. "Software Risk Management Practice in Industry and Support for Risk Engineering in the GOAL Toolset", in Proceedings of the Second SEI Conference on Software Risk Management SEI, Pittsburgh.

Roy, G. G. & Woodings, T. L. "A framework for risk analysis in software engineering", in Proceedings of the Seventh Asia-Pacific Software Engineering Conference IEEE, pp. 441-445.

Saaty, T. L. 1982, Decision Making for Leaders, Lifetime Learning Publications, Belmont, California.

Schach, Stephen R. 1999. Classical and object-oriented software engineering with UML and C++. Fourth Edition. Boston, McGraw-Hill.

SEI 1993, Proceedings of the Second SEI Conference on Software Risk Management, Software Engineering Institute, Pittsburgh, PA.

SEI 1994, Proceedings of the Third SEI Conference on Software Risk Management, Software Engineering Institute, Pittsburgh.

SEI 1995, Proceedings of the Fourth SEI Conference on Software Risk Management, Software Engineering Institute, Pittsburgh, PA.

SEI 1997, Proceedings of the Fifth SEI Conference on Software Risk Management, Software Engineering Institute, Pittsburgh, PA.

Singh, R. Information Technology Software Life-Cycle Process, ISO/IEC (JTC1)-SC7. 1991.

Sisti, F. J. & Joseph, S. 1994, Software Risk Evaluation Method Version 1.0, Software Engineering Institute, Pittsburgh, CMU/SEI-94-TR-19.

Sodhi, Jag. 1991. Software engineering: methods, management, and CASE tools. First Edition. McGraw-Hill.

Sommerville, I. 1996, Software Engineering, 5 edn, Addison-Wesley.

Speaker, W. V. "Implementing a Risk Management Methodology: Step 1 - Defining the Process", in Proceedings of the Second SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Stamatis, D. H. 1995, Failure Mode and Effect Analysis : FMEA from Theory to Execution, American Society for Quality.

Tversky, A. & Kahneman, D. 1974, "Judgment under Uncertainty: Heuristics and Biases", Science no. 185, pp. 1124-1131.

Van Scoy, Roger L. *Software Development Risk: Opportunity, Not Problem.* Software Engineering Institute, CMU/SEI-92-TR-30, ADA 258743, September 1992

Waller, R. A. & Covello, V. T. 1984, Low-Probability High-Consequence Risk Analysis, Plenum Press, New York.

Wang, J. X. & Roush, M. L. 2000, What Every Engineer Should Know About Risk Engineering and Management, Marcel Dekker.

Weyuker, E. J. "Predicting project risk from architecture", in Proceedings of the Sixth International Software Metrics Symposium pp. 82-90.

Whitten, N. 1995. Managing Software Development Projects – Formula for Success, Second Edition. New York, USA, John Wiley & Sons, Inc.

Williamson, J. A. "Experiences with an Independent Risk Assessment Team", in Proceedings of the Third SEI Conference on Software Risk Management SEI, Pittsburgh, PA.

Zahran, S. 1997. Minimizing the risk to software projects. The Quality Challenge, Vol 5.

Zahran, S. 1998. Software Process Improvement – Practical Guidelines for Business Success. Harlow, England, Addison-Wesley.

Anon. 1988, Software Risk Abatement, Department of the Airforce, Adrews Air Force Base, DC, 800-45.

Anon. 1992, The American Heritage Dictionary of the English Language, 3 edn, Microsoft Bookshelf/Houghton Mifflin Company, U.S.A.

Anon. 1995a, Merriam-Webster's Collegiate Dictionary, 10 edn, Merriam-Webster, Springfield, MA.

Anon. 2000b, Project management. Guide to the management of business related project risk, BS 6079-3:2000 edn, BSI.

10. APPENDIX

10.1 Appendix 1

Accepted risks	Description of risks, which exist but business owners have decided not to mitigate them. However, some of them will be actively followed
Action owner	Person who is responsible that risk-controlling actions are planned, agreed, documented and implemented.
Assurance	Nokia's approach to bring visibility and assurance to management and the Board on key risk areas and on the design and effectiveness of risk controls in business processes. Assurance means also the comfort the individual manager gets from risk management and control actions
Assessment	An analysis about the current state of processes and their maturity compared to defined benchmarks
Audit	Audit means an analysis which is done independently from the process/business owner in order to have an objective view on the audited areas by using defined methodology
Compliance	The status of the processes and controls compared to rules and regulations
Constraint	A limitation or rule that must be respected, e.g., " ... deadline is in September". Can be considered a type of "goal" i.e., a constraint needs to be protected like an objective.
Controlling action	A proactive maneuver that is taken before risk occurs (or before it is known whether the risk has occurred)

	known whether the risk has occurred)
Goal	Goal describes the intended results in question. Goals can be categorized into objectives, drivers, and constraints and identified at unit level
Goal and stakeholder review	A process step in risk management. The stated goals of the working entity (Business Unit, Product program, Business Group, Function etc.) are reviewed and refined. Stakeholders' associations with the goals are analyzed
Impact	The evaluated loss/effect in qualitative and monetary terms
Internal control	Internal control consists of measurement and monitoring of business objectives and performance, supported by quality, control, risk and monitoring processes
Maturity	Defines the level of understanding, framework and implementation of the issue in question
Objective	A goal that has an achievable, well-defined target level of achievement.
Probability	The likelihood of risk event occurrence by taking the current controls into account
Problem	Problem - a realized issue which can be defined as a consequence of a risk event identified at earlier stage
Risk	Any uncertainty that affects the objectives and achievement of optimum result.
Risk analysis	A process step in risk management. Risks are categorized and

Risk analysis	consolidated, risk scenarios are completed for main risk events and risk effects, probabilities and utility losses are estimated for all risk scenarios
Risk appetite	Risk appetite expresses the organizations willingness to take risks. Risk appetite is defined at the end by the board and can vary between risk areas and categories.
Risk audit	Risk based audit assessing the current status of risk management aspects in business processes
Risk categorizing	The act of grouping risks into sets that contain similar risk items
Risk control	A process step in risk management. Controlling actions are defined, decided upon, and implemented
Risk control strategy	A main control strategy that is used (Take, Treat, Terminate, Transfer) to manage the risk (accept or lower the probability and/or utility loss of risk scenarios)
Risk effect / consequence	The combined impact of risk event and resulting reactions to goals
Risk event	An occurrence of an incident with some negative consequences
Risk factor	A known fact or characteristic that influences some risk event
Risk identification	A process step in risk management. Potential threats to the project are identified using a chosen approach
Risk magnitude	Significance or size of the risk. Determined by a risk's probability of occurrence and (utility) loss would cause.

Risk management authority	Ultimate decision-making authority is defined in risk management mandate. Authority is to decide which approach to risk is to be taken in unclear situations.
Risk management coach	A Nokia person with professional skills and responsibilities in other areas but additionally authorized by his/her superiors and qualified for assisting the implementation of risk management activities within his/her respective area.
Risk management mandate	An explicit definition of the scope, frequency, focus, responsibility and chain of authorities in a specific area for risk management
Risk managementt mandate definition	A process step in risk management. The scope and frequency of risk management and the relevant stakeholders are defined in this step
Risk monitoring	A process step in risk management. Risk controlling actions and chosen/new risks are monitored in an agreed forum and interval.
Risk owner	Risk owner is the function or person who is responsible for managing the risk and takes the upside and downside to the P&L
Risk prioritization	Ranking of risks. Risk scenario probabilities are estimated and utility losses of scenarios are ranked separately for each relevant stakeholder
Risk scenario	A combination of risk elements that describe the causes, triggering events and the impact of a risk. Normally a scenario consists of a risk event, risk reaction and risk effect set
Risk tolerance	The grade of risk aversion defined by a company's ability to be exposed to various risks. Main criteria for defining risk tolerance are financial, operational and ethical measures

Root cause The original reason for risk to exist.

Stakeholder Any individual, group, organization, or institution who can affect or be affected by the work goals or their results

Self Assessment A concept designed for the organization and individuals to be able to define its own maturity and status at defined areas.

Utility loss The harm a stakeholder experiences on a set of risk effects in a situation

10.2 Appendix 2

ABC 1.0 Program Risk Log - Open Risks

Log Owner	Program Manager							
Next MS	E5							
Risk no	Description of risk	Raised date	Responsibility	Description of consequences	Probability	Planned preventive actions	Next action	Update date
1	Changing of software development model from waterfall model to iterative model	20.11.2003		delays in schedule	High			
2	Fierce competition in this field of mobile business lead to more competitive products from competitors	01.12.2003		lose the win-win cases due to release the product behind the competitors. Not enough attractive features.	Low			
3	This product is conceptualized as a current complement for the one already on the market for 4 years and in the future a substrute for it. This situation implies many interferences between the programs. However, the roadmap includes another two releases of the old product until the new one is mature	01.12.2003		delays in schedule; non-coordination/non-compatibility of the requirements bewteer complementary products; different release date	Medium			
4	The responsibility of one team of subcontractors (inside Nokia) passes to program	06.03.2004		delays in schedule	Fact			
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								

Program risks | R&D risks | Testing risks | old-Closed risks | old-Open Risks | old-Program top x

Figure 12. Risk log example

10.3 Appendix 3

RISK LOG TEMPLATE
By:
Entity / Area of business:

Date created:

Risk Activities	Activity Description	Risk 1	Risk 2
Area where the risk occurs	Requirements, Development, Testing, etc		
Root causes of risk event	Reasons why the risk event exists.		
Risk Event	Short description of the risk event endangering the achievement of the business objective(s).		
Consequences	Verbal description of what would it mean if the risk materializes		
Estimated Impact	Assess the costs of the damage caused by the consequences. Quantify the cost either in terms of time (schedule risks) or money (any kind of risks).		
Estimated impact scale (1-5)	1= Minor; 2= Some impact; 3= Meaningful impact, 4= Significant impact, 5= Major impact		
Estimated probability scale (1-3)	1= Low probability, 2= Medium probability, 3= High probability		
Risk magnitude	It can be calculated and estimated based on previous 2 rows values	N/A	N/A
Risk owner	Who should decide how to deal with this risk?		
Expected actions	Proposed or agreed actions to manage the risk.		
Action owner	Nominated person or the task force implementing the decided actions.		
Cost of corrective actions	Assess the cost of planned corrective actions.		
Current status of preventive actions	Describe current status of preventive actions		
Current risk status	Has the risk occurred?		
Schedule	Set schedule for the action(s).		
Escalation	Where the risk was escalated to and when?		
Information updated	When was the information of this risk updated?		
Current status for actions	Examples: - proposal - on going - pending - closed		

Figure 13. Risk log template – proposal